การพัฒนาแอพพลิเคชั่นบนกริด

GRID COMPUTING

นายปวัน นันทานิช
นายพงศ์ชยนต์ เอื้อประเสริฐ
นายอาทิตย์ ภูมิจิตอมร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

การพัฒนาแอพพลิเคชั่นบนกริด

GRID COMPUTING

นายปวัน นันทานิช

นายพงศ์ชยนต์ เอื้อประเสริฐ

นายอาทิตย์ ภูมิจิตอมร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาแอพพลิเคชั่นบนกริด

Grid Computing

ผู้จัดทำ

1. นายปวัน นันทานิช รหัสนักศึกษา 46010437
2. นายพงศ์ชยนต์ เอื้อประเสริฐ รหัสนักศึกษา 46010471
3. นายอาทิตย์ ภูมิจิตอมร รหัสนักศึกษา 46010961

อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโกคา)

การพัฒนาแอพพลิเคชั่นบนกริด

นายปวัน นันทานิช 46010437
นายพงศ์ชยนต์ เอื้อประเสริฐ 46010471
นายอาทิตย์ ภูมิจิตรอมร 46010961
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

ความต้องการทรัพยากรในการประมวลของคอมพิวเตอร์ซึ่งมากขึ้นเรื่อยๆในปัจจุบัน เช่น โครงการ SETI@Home ที่เป็นโครงการที่วิจัยเกี่ยวกับสัญญาณที่รับได้จากนอกโลกผ่านทางกล้องโทรทรรศน์ขนาดใหญ่ ซึ่งเป็นข้อมูลจำนวนมหาศาลมากกว่า Terabytes ในแต่ละวันที่ต้องการการประมวลผล จึงมีเทคโนโลยี Grid Computing เกิดขึ้น ซึ่งเป็นเทคโนโลยีที่กลุ่มผู้ร่วมงานได้เลือกมาเป็นหัวข้อโครงงาน เพื่อนำมาศึกษาและพัฒนาต่อให้สามารถนำไปใช้ประโยชน์ในด้านต่างๆไม่ทางใดก็ทางหนึ่ง โดยสิ่งที่ทำในโครงงานนี้คือ การส่งงานมัลติมีเดียไปเข้ารหัสบนเครื่องที่อยู่ในระบบ Grid และ Cluster นำมาทำเป็นกริดแอพพลิเคชัน โดยสิ่งที่ต้องทำในโครงงานนี้คือการศึกษาการใช้ Globus Toolkit, PBS (Portable Batch System) และองค์ประกอบอื่นๆ เช่น MDS (Monitoring and Discovering System) และ Shell Script เพื่อนำมาสร้างเป็นกริดแอพพลิเคชันส่งงานการเข้ารหัสไฟล์มัลติมีเดียร์บนเครื่องในระบบกริดให้สามารถทำงานได้ อีกทั้งยังใช้ Shell Script เพื่อตรวจสอบว่าเครื่องที่จะส่งงานไปนั้นว่างอยู่หรือไม่โดยที่เครื่องที่อยู่ในระบบกริดจะใช้ GRAM ช่วยในการตรวจสอบว่างานเสร็จแล้วหรือไม่ ส่วนเครื่องที่อยู่ในระบบ Cluster จะใช้ shell script ตรวจสอบว่าเครื่องใดมี CPU Load น้อยกว่าและตรวจสอบว่างานเสร็จแล้วหรือไม่ เพื่อให้ทำงานได้อย่างมีประสิทธิภาพ

Grid Computing

Mr. Pawan Nunthanid 46010437
Mr. Pongshayon Uerprasert 46010471
Mr. Arthit Phumchitamorn 46010961
Dr. Voravat Limpoka Advisor
Academic Year 2006

ABSTRACT

The requirement of environment for computing is more increase today ,like SETI@Home project that the object of this project is searching about signal from outer worlds through huge telescope.It has data more than terabytes each day. So grid computing technology will be used to solve it.This thesis select grid computing technology to learn and develop. In this project we use grid for sending multimedia files to encode on each station in Grid and Cluster system. Studying Globus toolkit, PBS and other component, like MDS (Monitoring and Discovering System) and Shell Script, build grid application for sending multimedia job to encode on each station in grid system by using GRAM to examine which job is completed. Each station in Cluster use shell script to examine which station has less CPU Load than others and job is completed.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้เป็นอย่างดีด้วยคำแนะนำและคำปรึกษาจาก ดร.วรวัฒน์ ลิ้มโภคา ซึ่งเป็นอาจารย์ผู้ควบคุมปริญญานิพนธ์ ข้าพเจ้ารู้สึกซาบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอกราบขอบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกๆท่าน ที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณพี่ๆที่ไทยกริดที่สละเวลาอันมีค่า มาให้คำปรึกษาและแนะนำข้อมูลเชิงเทคนิคต่างๆให้กับข้าพเจ้า

ขอขอบคุณเพื่อนๆ ทุกคนในห้อง Olala ที่คอยร่วมทุกข์ร่วมสุขอยู่ทำ Project กันมาทั้งปี รวมทั้งคอยให้คำแนะนำและคำปรึกษาเสมอมา

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนในทุกๆเรื่อง ทำให้ข้าพเจ้าสามารถทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากปริญญานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

นายปวัน นันทานิช
นายพงศ์ชยนต์ เอื้อประเสริฐ
นายอาทิตย์ ภูมิจิตอมร

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

โครงงานกริดฉบับนี้เป็นโครงงานในปีการศึกษา 2549 ซึ่งเป็นโครงงานที่ทำต่อเนื่องจากโครงงานในปีการศึกษา 2548 โดยสิ่งที่ได้ทำในปีการศึกษา 2548 นั้น ได้ใช้กระบวนการของเครื่องมือทางกริด Globus toolkit 3.2.1 ในการกระจายงานการคูณเมตริกซ์ที่มีมิติขนาดใหญ่ โดยมีขั้นตอนการสร้างแอพพลิเคชั่น โดยเริ่มจากการเขียนกริดเซอร์วิส ซึ่งมีขั้นตอนต่างๆ คือ กำหนดรูปแบบของเซอร์วิสด้วย GWSDL แล้วทำการสร้างเซอร์วิสด้วยภาษาจาวา จากนั้นทำการกำหนดค่าของส่วนที่จะนำมาใช้งานด้วย WSDD แล้วสร้าง GAR ไฟล์โดยการคอมไพล์ทุกส่วนนำมารวมกันซึ่งใช้ Ant ช่วย จากนั้นจึงบรรจุกริดเซอร์วิสไว้ในคอนเทนเนอร์ ซึ่งในโครงงานได้บรรจุ Matrix Service ลงไปในคอนเทนเนอร์ แล้วเขียนจาวาแอพพลิเคชั่นในการกระจายงาน โดยการเรียก Matrix Service แบบวนลูปเพื่อส่งงานคำนวณไปยังเครื่องในระบบ

เนื่องจากการกระจายงานที่ได้กล่าวมายังไม่ได้ใช้องค์ประกอบที่มีอยู่ของ Gloubus toolkit อย่างชัดเจนและไม่ได้มีการตรวจสอบงานที่ทำการส่งไปว่ามีสถานะเป็นอย่างไร เป็นการวนลูปเพื่อส่งงานคำนวณไปเรื่อยๆเรียงตามลำดับเครื่องในระบบ ดังนั้น ในโครงงานปีการศึกษา 2549 นี้ จึงได้ทำการศึกษาองค์ประกอบของ Globus toolkit 4.0.1 ซึ่งอัพเดตเวอร์ชั่นจาก Globus toolkit 3.2.1 แล้วนำแต่ละองค์ประกอบนั้นมาใช้งาน โดยในโครงงานจะสร้างระบบในเครือข่ายกริดมีจำนวน 3 เครื่อง ลงระบบแล้วเขียนแอพพลิเคชั่นในการส่งกระจายงานในการเข้ารหัสมัลติมีเดียร์ไฟล์ให้เป็น MPEG4 โดยนำโปรแกรม mplayer mencoder ซึ่งเป็นโปรแกรม Open source ในการเข้ารหัสและตัดต่อมัลติมีเดียร์ไฟล์เข้ามาช่วยและทำการเพิ่มส่วนที่เป็นคลัสเตอร์สำหรับระบบกริดจำนวน 2 เครื่อง โดยใช้ TORQUE ซึ่งเป็นเครื่องมือจัดการระบบคลัสเตอร์ โดยเป็นคลัสเตอร์แบบ PBS (Portable Batch System)

1.1. วัตถุประสงค์ของโครงงาน

1.1.1. เพื่อศึกษาองค์ประกอบของ Globus Toolkit รวมไปถึงเครื่องมือทางคลัสเตอร์ PBS และสามารถนำมาใช้งาน

1.1.2. เพื่อศึกษาแนวคิดและการประมวลผลและการสร้างระบบกริด

1.1.3. เพื่อศึกษาและออกแบบแอพพลิเคชั่นในการกระจายงานที่เหมาะสมในระบบกริด

1.2. ขอบเขตของโครงงาน

โครงงานนี้มีจะพัฒนาแอพพลิเคชั่นกระจายงานในการเข้ารหัสมัลติมีเดียร์ไฟล์ โดยจะทำการศึกษาและนำเครื่องมือทางกริดมาใช้ให้เกิดประโยชน์สูงสุด ซึ่งจะนำองค์ประกอบหลักของ Globus Toolkit 4.0.1 ทั้ง 3 ส่วน คือ GRAM, MDS และ GridFTP มาใช้งานรวมไปถึงการนำเครื่องมือ TORQUE ที่เป็นเครื่องมือจัดการระบบคลัสเตอร์สำหรับกริดมาศึกษาและประยุกต์ใช้ในโครงงาน

1.3. วิธีการดำเนินงาน

1.3.1. ศึกษาค้นคว้าทฤษฎีขององค์ประกอบหลักของ Globus toolkit 4.0.1

1.3.2. ติดตั้งและเตรียมสภาพแวดล้อมของซอฟแวร์ที่ต้องใช้ร่วมกับ Globus Toolkit เพื่อให้แอพพลิเคชั่นที่ต้องการพัฒนาสามารถทำงานได้

1.3.3. เนื่องจากมีสมาชิกทำงาน 3 คน จึงแบ่งกลุ่มงานออกเป็น 3 กลุ่ม

1.3.4. คนแรกทดลองเขียนลักษณะงาน (Job description) เพื่อใช้ส่งเข้าประมวลผล โดยใช้ภาษา RSL ในรูปแบบ XML ซึ่งเป็นรูปแบบการใช้ GRAM และ GridFTP ซึ่งเกี่ยวเนื่องกัน

1.3.5. คนที่สองศึกษาวิธีการใช้งาน MDS ที่ใช้ในการจัดหาข้อมูลของทรัพยากรต่างๆ ในระบบกริด

1.3.6. คนที่สามศึกษาการเขียน Shell script เพื่อควบคุมการส่งงานให้เป็นแบบ Dynamic

1.3.7. ทดลองใช้โปรแกรม mencoder ที่ใช้สำหรับตัดต่อและเข้ารหัสมัลติมีเดียร์ไฟล์

1.3.8. เขียนลักษณะงานของไฟล์ที่จะเข้ารหัสแต่ละไฟล์แล้วทดลองส่งงานเข้าประมวลผล

1.3.9. ทดลองเขียนแอพพลิเคชั่นเบื้องต้นโดยใช้ Shell script ควบคุมการกระจายงาน

1.3.10. ทดสอบการทำงานของแอพพลิเคชั่นที่สร้างขึ้น

1.3.11. นำเครื่องมือทางคลัสเตอร์เข้ามาทดลองประกอบเข้ากับระบบแล้วทดลองใช้งานร่วมกัน

1.3.12. ประเมินผลการทดลอง

1.4. ประโยชน์ที่คาดว่าจะได้รับ

1.4.1. ได้รับความรู้ ความเข้าใจแนวคิดของการประมวลผลแบบกริด

1.4.2. ได้รับความรู้ ความเข้าใจเกี่ยวกับกระบวนการการแตกงานออกเป็นส่วนๆ เพื่อร่วมกันประมวลผลภายในกริด

1.4.3. ได้รับความรู้ ความเข้าใจเกี่ยวกับโครงสร้างและลักษณะการทำงานของกริด

1.4.4. สามารถสร้างระบบกริดขึ้นเพื่อแสดงถึงความแตกต่างระหว่างระบบประมวลผลทั่วไปกับระบบประมวลผลแบบกริด

1.5. เนื้อหาของโครงงาน

วิทยานิพนธ์ฉบับนี้จะประกอบด้วยความรู้เกี่ยวกับองค์ประกอบของเครื่องมือทางกริดเป็นส่วนใหญ่ เนื่องจากต้องใช้เวลามากในการศึกษาและการใช้งาน ส่วนประกอบอื่นๆของเนื้อหาจะเป็นส่วนของทฤษฎีที่ใช้ในกริดเทคโนโลยี ลักษณะโครงสร้างของกริด ลักษณะการทำงานของกริด การทำงานของ scheduler ในส่วนของคลัสเตอร์ที่เปรียบเสมือน pooling resource ของระบบกริดโดยใช้ PBS scheduler ภายใต้เครื่องมือ TORQUE และในส่วนสุดท้ายจะเป็นวิธีการสร้างแอพพลิเคชั่นและการกระจายงานในการเข้ารหัสไฟล์มัลติมีเดียร์ รวมไปถึงผลของการทดสอบการทำงาน

อนึ่ง ในการพัฒนาโครงงานชิ้นนี้ได้มีการศึกษาถึงการออกแบบและทดลองสร้างระบบกริดให้สามารถใช้งานได้จริง รวมถึงสามารถพัฒนาในส่วนของการเขียนโปรแกรมให้สามารถใช้งานบนระบบที่สร้างขึ้นได้ โดยไม่ได้เน้นในส่วนของการรักษาความปลอดภัยของระบบมากนัก โดยจะใช้เพียงแค่ระบบการรักษาความปลอดภัยเท่าที่ Globus Toolkit มีให้เท่านั้น ซึ่งถ้าผู้สนใจจะนำระบบที่ผู้จัดทำไปใช้งานจริงอาจเจอปัญหาในเรื่องของความปลอดภัยบ้าง ก็สามารถเพิ่มเติมส่วนของระบบความปลอดภัยเพิ่มเข้าไปได้ ซึ่งในส่วนนี้ทางคณะผู้จัดทำจะกล่าวถึงคร่าวๆในส่วนของทฤษฎีที่เกี่ยวข้อง

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในโครงงาน

2.1. มาตรฐาน OGSA และ OGSI

OGSA (Open Grid Services Architecture) เปรียบเสมือนพิมพ์เขียวในการที่จะสร้างมิดเดิ้ลแวร์ขึ้นมา หรือสามารถกล่าวได้ว่า OGSA เป็นโปรโตคอลพื้นฐานที่กำหนดวิธีการการติดต่อ,การส่งข้อมูล, การจัดเตรียมรีซอร์ส, การแชร์รีซอร์ส, การทำตารางเวลา และอื่นๆที่เกี่ยวข้อง โดยอาศัยเทคโนโลยีอยู่ 2 อย่างในการสร้างมันขึ้นมา คือ เครื่องมือ และ เว็บเซอร์วิส โดยสร้างขึ้นมาเป็น Open Grid Service Infrastructure (OGSI) ที่จะนำมาใช้งานจริง ในส่วนของ OGSA นี้พัฒนาโดย The Globus Grid Forum เน้นการกำหนกมาตรฐานและโครงสร้างสำหรับแอพพลิเคชั่นพื้นฐานของกริด เป็นมาตรฐานที่กำหนด Grid Service ซึ่งเป็นเทคโนโลยีที่แฝงอยู่ใน OGSA

OGSI (Open Grid Services Infrastructure) จะเป็นวิธีการในการสร้างกริดเซอร์วิส ซึ่งจะหมายถึงการจัดการ การแลกเปลี่ยนข้อมูลระหว่าง ผู้ใช้งานกริด และ เซอร์วิส อื่นๆ หรืออาจพูดได้ว่า เป็นเว็บเซอร์วิสที่จะช่วยเพิ่มความสะดวกให้กับการทำระบบกริด OGSI เป็นอีกมาตรฐานหนึ่งที่สร้างโดย OGSA ซึ่งเป็นมาตรฐานที่กำหนดข้อมูลเฉพาะเชิงเทคนิคว่า Grid Service นั้นคืออะไรในมุมมองของโครงสร้างชั้นสูง

2.2. ประเภทของกริด

2.2.1. Computational Grid

Computational Grid จะเน้นไปที่กำลังในการประมวลผล เครื่องที่ใช้มักจะเป็นเครื่องเซิร์ฟเวอร์ที่มีประสิทธิภาพสูง

ตัวอย่างของ Computational Grid ที่รู้จักกันดีคือ SETI@home กริดชนิดนี้ประกอบด้วยคอมพิวเตอร์ที่มีกำลังต่ำ และความจุข้อมูลน้อย โดย CPU cycle ที่ว่างของเครื่องคอมพิวเตอร์ส่วนบุคคลใน SETI@home ถูกรวมเข้าด้วยกันเพื่อสร้าง Computational Grid ที่ใช้วิเคราะห์คลื่นวิทยุที่ได้รับจากนอกโลกเพื่อค้นหาเซาว์ปัญญาจากโลกอื่น

องค์กรส่วนใหญ่ที่ใช้ Computational Grid เกิดจากมีการริเริ่มทางธุรกิจในการขยายความสามารถและเพิ่มการใช้ทรัพยากรที่มีอยู่ของคอมพิวเตอร์โดยการรวมและแชร์ทรัพยากร ในธุรกิจอาจจะต้องการความจุของคอมพิวเตอร์มากกว่าจำนวนที่มีอยู่จึงมีธุรกิจที่สนใจในการดัดแปลงแอพพลิเคชั่นเพื่อใช้การประมวลผลแบบขนาน

การใช้ Computational Grid รวมถึงการคิดสมการทางคณิตศาสตร์, การคิดราคา, การประเมินค่า portfolio และการจำลองแบบ (โดยเฉพาะการวัดความเสี่ยง) โดยไม่ใช่ทุกอัลกอริทึมที่จะสามารถใช้การประมวลผลแบบขนานได้

2.2.2. Scavenging Grid

Scavenging Grid เกิดจากการรวมกันของเครื่อง Desktop จำนวนมาก โดยเป็นการรวม CPU cycle และทรัพยากรอื่นๆ ซึ่งจะมีเครื่องคอมพิวเตอร์ 1 เครื่องที่ทำหน้าที่ควบคุมทรัพยากรที่มีอยู่ในกริดให้กับเครื่องคอมพิวเตอร์เครื่องอื่นๆที่ทำงานร่วมกัน

2.2.3. Data Grid

Data Grid ดูแลและควบคุมการเข้าถึงข้อมูลในหลายองค์กร โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ตำแหน่งที่อยู่ของข้อมูลตลอดระยะเวลาที่ใช้ข้อมูล ตัวอย่างเช่น มหาวิทยาลัยสองแห่งทำการวิจัยวิทยาศาสตร์ชีวภาพโดยใช้ข้อมูลเดียวกัน Data Grid จะช่วยในการแชร์ข้อมูล, จัดการข้อมูล, รักษาความปลอดภัยอย่างเช่น กำหนดว่าผู้ใช้แต่ละคนสามารถเข้าไปใช้ข้อมูลส่วนไหนได้บ้าง, และกำหนดการเข้าถึงข้อมูลที่น่าเชื่อถือและรวดเร็วธุรกิจที่ใช้ Data Grid มักจะเกิดจากการริเริ่มต้นทางสารสนเทศเพื่อขยายความ สามารถของเหมืองแร่ข้อมูล และเพิ่มประโยชน์ในการลงทุนโครงสร้างพื้นฐานในการเก็บข้อมูลที่มีอยู่และลดความซับซ้อนในการจัดการข้อมูล

2.3. ลักษณะของความปลอดภัยที่ใช้ในกริด

ในส่วนของความปลอดภัยภายในกริดจะใช้เทคนิคการเข้ารหัสแบบ public and private key cryptography ซึ่งอยู่ในขั้นตอนของการทำ simpleCA โดยการ sign certificate ให้ผู้ใช้ต่างๆ ในระบบที่ยื่นคำร้องเข้ามาขอใช้งานเซอร์วิสของกริดที่บรรจุอยู่ในคอนเทนเนอร์ โดยผู้ใช้ทั้งหมดจะต้องได้รับการ sign certificate ก่อนการเข้ามาใช้งาน เนื้อหาในส่วนนี้จึงทำขึ้นเพื่อให้เข้าใจรายละเอียดของการเข้ารหัสลับรูปแบบนี้ได้ดียิ่งขึ้น

2.3.1. นิยามความมั่นคงปลอดภัยคอมพิวเตอร์

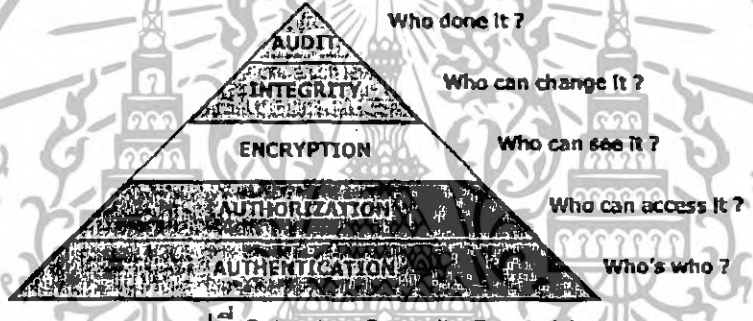
ในปัจจุบันระบบคอมพิวเตอร์ได้ถูกคุกคามมากขึ้น ทั้งจากไวรัสคอมพิวเตอร์หรือจากผู้ไม่ประสงค์ดี ซึ่งความมั่นคงปลอดภัยคอมพิวเตอร์ (Computer Security) ช่วยปกป้องเครื่องคอมพิวเตอร์รวมไปถึงอุปกรณ์ต่างๆที่เกี่ยวข้อง และที่สำคัญยังสามารถช่วยปกป้องข้อมูลที่ได้จัดเก็บไว้ภายในระบบหรือใช้ในความหมายความปลอดภัยทางข้อมูลสารสนเทศ (Information Security) ก็ได้ จุดประสงค์หลักของความปลอดภัยทางข้อมูลคือ ความลับ (Confidentiality) ความสมบูรณ์ (Integrity) ความพร้อมใช้ (Availability) และการห้ามปฏิเสธความรับผิดชอบ (Non-Repudiation) ของข้อมูลต่างๆภายในองค์กร (CIA-N) โดยมีรายละเอียดดังนี้

2.3.1.1. **การรักษาความลับ** (Confidentiality) คือการรับรองว่าจะมีการเก็บข้อมูลไว้เป็นความลับ และผู้มีสิทธิเท่านั้นจึงจะเข้าถึงข้อมูลนั้นได้

2.3.1.2. **การรักษาความสมบูรณ์** (Integrity) คือ การรับรองว่าข้อมูลจะไม่ถูกเปลี่ยนแปลงหรือทำลายไม่ว่าจะเป็นโดย อุบัติเหตุหรือโดยเจตนา

2.3.1.3. **ความพร้อมใช้** (Availability) คือการรับรองว่าข้อมูลและบริการการสื่อสารต่าง ๆ พร้อมที่จะใช้ได้ในเวลาที่ต้องการใช้งาน

2.3.1.4. **การห้ามปฏิเสธความรับผิดชอบ** (Non-Repudiation) คือวิธีการสื่อสารซึ่งผู้ส่งข้อมูลได้รับหลักฐานว่าได้มีการส่งข้อมูลแล้วและผู้รับก็ได้รับการยืนยันว่าผู้ส่งเป็นใคร ดังนั้นทั้งผู้ส่งและผู้รับจะไม่สามารถปฏิเสธได้ว่าไม่มีความเกี่ยวข้องกับข้อมูลดังกล่าวในภายหลัง ในทางปฏิบัตินั้นสามารถกำหนดลักษณะของการควบคุมความมั่นคงปลอดภัย (Security Controls) ได้ 5 ระดับตามรูป



รูปที่ 2.1 แสดง Security Pyramid

ถือเป็นองค์ประกอบที่สำคัญส่วนหนึ่งของความมั่นคงปลอดภัยคอมพิวเตอร์ เพราะจัดเป็นการกำหนดและควบคุมทั้งบุคคลที่สามารถเข้าสู่ระบบและเข้าสู่ข้อมูลภายในระบบ และเพื่อกระทำการใดได้บ้าง อนุญาตตามระดับขั้นของความสำคัญของข้อมูล รวมไปถึงการจัดเก็บพฤติกรรมการใช้งานระบบของบุคคลนั้นต่อข้อมูลบนระบบทั้งหมด

2.3.2. **การพิสูจน์ตัวตนโดยการเข้ารหัสโดยใช้กุญแจสาธารณะ** (Public-key cryptography)

เป็นการรักษาความปลอดภัยของข้อมูลระหว่างการส่งข้ามเครือข่ายวิธีหนึ่งที่นิยมใช้กันอยู่ในปัจจุบัน การเข้ารหัสแบบคู่รหัสกุญแจนี้จะมีความปลอดภัยมากกว่าการเข้ารหัสข้อมูลแบบธรรมดา แต่ก็ไม่ได้หมายความว่าการเข้ารหัสแบบคู่รหัสกุญแจนี้จะเป็นวิธีที่เหมาะสมที่สุดของวิธีการเข้ารหัส ทั้งนี้ขึ้นอยู่กับประเภทงานของแต่ละองค์กรหรือบุคคล

การเข้ารหัสโดยใช้กุญแจสาธารณะ ประกอบไปด้วยกุญแจ 2 ชนิด ที่ต้องใช้คู่กันเสมอในการเข้ารหัสและถอดรหัสคือ

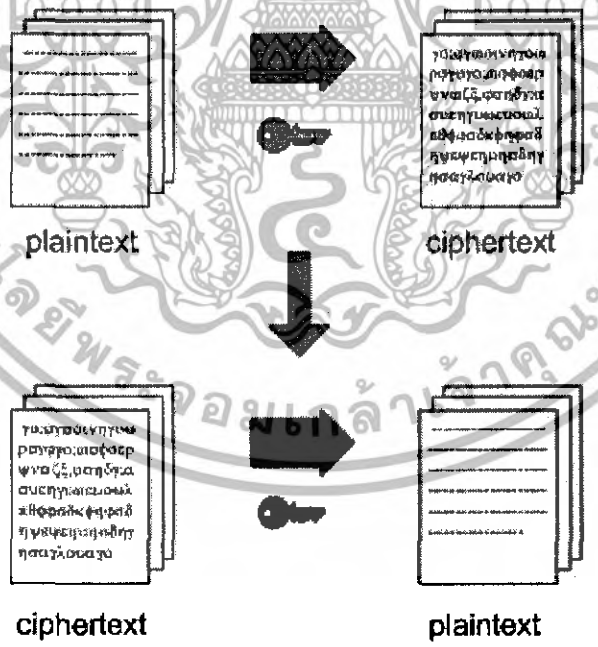
2.3.2.1. **กุญแจสาธารณะ** (public-key) เป็นกุญแจที่ผู้สร้างจะส่งออกไปให้ผู้ใช้อื่นๆ ทราบหรือเปิดเผยได้

2.3.2.2. **กุญแจส่วนตัว** (private-key) เป็นกุญแจที่ผู้สร้างจะเก็บไว้ โดยไม่เปิดเผยให้คนอื่นรู้ กระบวนการของการเข้ารหัสแบบคู่รหัสกุญแจ มีดังนี้

ผู้ใช้แต่ละคนจะสร้างคู่รหัสกุญแจของตัวเองขึ้นมา เพื่อใช้สำหรับการเข้ารหัสและการถอดรหัส กุญแจสาธารณะจะถูกส่งออกไปยังผู้ใช้คนอื่นๆ แต่กุญแจส่วนตัวจะถูกเก็บที่ตนเอง เมื่อจะส่งข้อมูลออกไปหาผู้ใช้คนใด ข้อมูลที่ส่งจะถูกเข้ารหัสด้วยกุญแจสาธารณะ ก่อนถูกส่งออกไป เมื่อผู้รับได้รับข้อความแล้วจะใช้กุญแจส่วนตัวซึ่งเป็นคู่รหัสกันถอดรหัสออกมา

การเข้ารหัสโดยใช้กุญแจสาธารณะสามารถใช้ได้ทั้งในการเข้ารหัส (Encryption) และการพิสูจน์ตัวตน (Authentication)

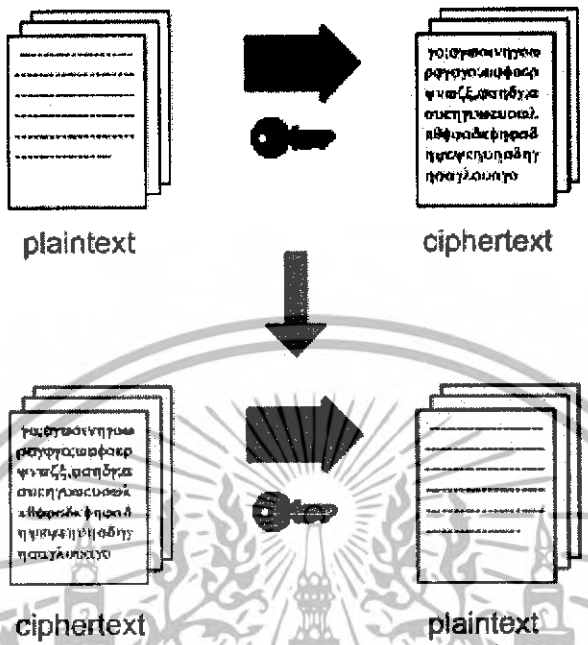
การประยุกต์ใช้ในการเข้ารหัสข้อมูล (Encryption) เป็นการนำข้อมูลที่จะส่งไปยังผู้รับมาเข้ารหัสด้วยกุญแจสาธารณะของผู้รับ และเมื่อผู้รับได้รับข้อความนั้นแล้วจะถอดรหัสออกมาด้วยกุญแจส่วนตัว จึงจะเห็นได้ว่ามีเพียงผู้รับเท่านั้นที่จะสามารถถอดรหัสออกมาได้



รูปที่ 2.2 ระบบของการเข้ารหัสแบบใช้คู่รหัสกุญแจ

การประยุกต์ใช้ในการพิสูจน์ตัวตน (Authentication) เป็นการนำข้อมูลที่ผู้ส่งต้องการส่งมาเข้ารหัสด้วยกุญแจส่วนตัวของผู้ส่ง แล้วนำข้อมูลนั้นส่งไปยังผู้รับ ซึ่งผู้รับจะใช้กุญแจสาธารณะ

ซึ่งเป็นคู่รหัสกันถอดรหัสออกมา ผู้รับก็สามารถรู้ได้ว่าข้อความนั้นถูกส่งมาจากผู้ส่งคนนั้นจริง ถ้าสามารถถอดรหัสข้อมูลได้อย่างถูกต้อง



รูปที่ 2.3 ระบบของการเข้ารหัสแบบใช้คู่รหัสกุญแจเพื่อการพิสูจน์ตัวตน

2.3.3. **การพิสูจน์ตัวตนโดยการใช้ลายเซ็นอิเล็กทรอนิคส์** (Digital Signature)

เป็นการนำหลักการของการทำงานของระบบการเข้ารหัสแบบใช้คู่รหัสกุญแจ เพื่อการพิสูจน์ตัวตนมาประยุกต์ใช้ ระบบของลายเซ็นดิจิตอลสามารถแบ่งเป็นขั้นตอนได้ดังนี้

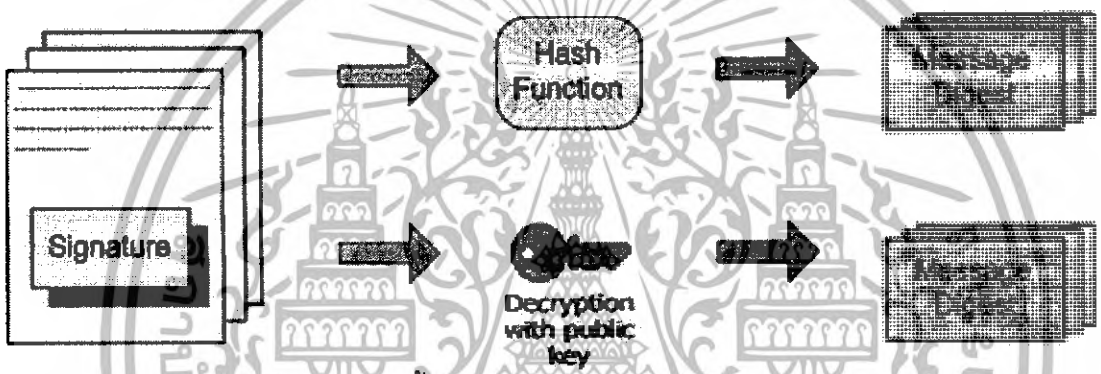
2.3.3.1. เมื่อผู้ใช้ต้องการจะส่งข้อมูลไปยังผู้รับ ข้อมูลนั้นจะถูกนำไปเข้าฟังก์ชั่นทางคณิตศาสตร์ที่เรียกว่า "แฮชฟังก์ชัน" ได้เมสเซสไดเจสต์ (Message Digest) ออกมา

รูปที่ 2.4 การส่งข้อมูลเข้าไปใน Hash function

2.3.3.2. การใช้กุญแจส่วนตัวเข้ารหัสข้อมูล หมายถึงว่าผู้ส่งได้ลงลายเซ็นดิจิตอล ยินยอมที่จะให้ผู้รับ สามารถทำการตรวจสอบด้วยกุญแจสาธารณะของผู้ส่งเพื่อพิสูจน์ตัวตนของผู้ส่งได้

รูปที่ 2.5 การเข้ารหัสเมสเซสไดเจสต์ด้วยกุญแจส่วนตัวเพื่อเป็นการลงลายเซ็น

2.3.3.3. การตรวจสอบข้อมูลว่าถูกส่งมาจากผู้ส่งคนนั้นจริงในด้านผู้รับ โดยการนำข้อมูลมาผ่านแฮชฟังก์ชันเพื่อคำนวณหาค่าเมสเซจไดเจสต์ และถอดรหัสลายเซ็นอิเล็กทรอนิคส์ด้วยกุญแจสาธารณะของผู้ส่ง ถ้าสามารถถอดได้อย่างถูกต้อง จะเป็นการยืนยันข้อมูลจากผู้ส่งคนนั้นจริง และถ้าข้อมูลเมสเซจไดเจสต์ที่ได้จากการถอดรหัสเท่ากันกับค่าเมสเซจไดเจสต์ในตอนต้นที่ทำการคำนวณได้ จะถือว่าข้อมูลดังกล่าวนั้นถูกต้อง



รูปที่ 2.6 ขั้นตอนการเปรียบเทียบความถูกต้อง

ลายเซ็นอิเล็กทรอนิคส์ นิยมนำไปใช้ในระบบรักษาความปลอดภัยในการชำระเงินผ่านระบบอินเตอร์เนต ซึ่งในปัจจุบันนี้การทำธุรกรรมการเงินอิเล็กทรอนิกส์ได้รับความนิยมเป็นอย่างมาก

2.4. Scheduling

Scheduling เป็นการจัดการสำคัญที่ใช้ในระบบปฏิบัติการ เพื่อการจัดการ resource ต่างๆ เนื่องจาก resource มีจำกัด และ resource เกือบทุกชนิด ในระบบต้องใช้ร่วมกัน ดังนั้นต้องมีวิธีการจัดลำดับการเข้าใช้ resource ของ process ต่างๆ เสมอ วิธีการจัดลำดับที่จะได้ศึกษาครั้งแรก คือ การจัดลำดับให้ process เข้า ใช้ CPU ซึ่งเป็น resource สำคัญของระบบปฏิบัติการ

Basic concept วัตถุประสงค์ในการจัดการของ multiprogramming mechanism คือต้องมี process กำลังใช้ CPU ในขณะหนึ่ง ๆ เสมอ เพื่อทำให้เกิด CPU Uitlisation สูงสุด อย่างไร

ก็ตามในสิ่งแวดล้อมแบบ uniprocess system จะมีเพียงหนึ่ง process เท่านั้น ที่สามารถใช้ CPU ได้ในขณะหนึ่ง ๆ และ process อื่น ๆ ต้องรอ เพื่อเข้าใช้ CPU ต่อเมื่อ CPU ว่างลง

ในการทำงานของระบบ multiprogramming จะต้องมีการกระจายโอกาสในการเข้าใช้ CPU ให้แก่หลายๆ process ซึ่งโดยทั่วไปจะจัดให้มี process หลายๆ process ใน memory เพื่อสามารถเข้าใช้ CPU ได้เมื่อ CPU ว่างลง (เมื่อ process ที่กำลังใช้ CPU เสร็จการทำงาน หรือขอใช้ I/O) อย่างไรก็ตามเนื่องจากในขณะใดๆ จะมีเพียง process เดียวเท่านั้น ที่สามารถเข้าใช้ CPU ได้ ดังนั้นจะต้องมีวิธีการจัดลำดับ การได้เข้าถึง CPU ของ process ที่รอ ใน memory ซึ่งการทำงานในส่วนนี้คือ หน้าที่ของ CPU Scheduler

วิธีการจัดการของ CPU Scheduler จะเป็นตัวบ่งชี้หรือดัชนีบอกถึง ระดับของ CPU Utilisation และประสิทธิภาพโดยรวมของระบบด้วย ดังนั้นการจัดการของ CPU Scheduler จะต้องคำนึงถึงส่วนประกอบหลายอย่าง เพื่อก่อให้เกิดประสิทธิภาพสูงสุด

2.4.1. First-Come, First-Served Scheduling (FCFS)

FCFS เป็น Algorithm ที่ง่ายที่สุดในการจัด cpu ให้แก่ process (หรือจัด process เข้าใช้ cpu) โดยจัดตามลำดับก่อนหลังของ process ที่อยู่ใน ready queue

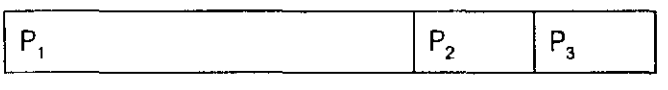
Average waiting time ของ FCFS จะสูงเมื่อเทียบกับการจัดการในรูปแบบอื่น ซึ่ง average waiting time ของ process คือค่าเฉลี่ยของเวลาที่แต่ละ process ต้องรอเพื่อเข้าใช้ cpu นั่นเอง

ตัวอย่างการคำนวณค่า average waiting time ของ FCFS Algorithm

ถ้าพิจารณาจาก process P_1 P_2 P_3 ซึ่งมี cpu burst time ดังกำหนดให้จากตารางคือ

Process	CPU Burst Time(milliseconds)
P_1	24
P_2	3
P_3	3

ถ้าลำดับของการเข้าใช้ cpu คือ P_1 P_2 P_3 ตามลำดับคือ



เวลา 0 24 27 30

P_1 ต้องรอ 0 milliseconds

P_2 ต้องรอ 24 milliseconds

P_3 ต้องรอ 27 milliseconds

ดังนั้น Average waiting time = (0+24+27)/3 = 17 milliseconds

ถ้าลำดับการเข้าสู่ ready queue ของ process ทั้งสามนี้เปลี่ยนไปเป็น (เช่นดังตัวอย่าง) $P_2 P_3$ และ P_1 จะได้

เวลา 0 3 6 30

จะได้ว่า

Waiting time = (0+3+6)/3 = 3 milliseconds

ดังนั้นจะเห็นว่า average waiting time ของ FCFS policy ขึ้นอยู่กับลำดับการเข้าสู่ ready queue และ cpu burst time ของแต่ละ process ด้วย

ปัญหา ที่อาจจะเกิดขึ้นได้ในกรณีที่มี cpu bounded process กำลังใช้ cpu อยู่และมีหลาย I/O bounded process รอใช้ cpu ในขณะหนึ่งๆเนื่องจาก cpu bounded process ใช้เวลานานในการใช้ cpu ทำให้ process อื่นๆ ที่รอ I/O operation ทั้งหมดหรือส่วนใหญ่พร้อมที่จะเข้าใช้ cpu ต่อ (I/O operation complete) ทำให้ process ส่วนใหญ่ต้องรอใน ready queue (queue นี้จะยาว) และขณะเดียวกันจะไม่มี process ไหนขอใช้ I/O ได้จนกว่าจะได้กลับเข้าใช้ cpu อีกครั้งทำให้เกิดการว่างงานของ I/O device

ในที่สุดเมื่อ cpu bounded process สิ้นสุด cpu burst cycle และกลับสู่การร้องขอ I/O device (เพื่อใช้ I/O) process ที่อยู่ใน ready queue ได้รับการจัดสรรเข้าสู่ cpu แต่ใช้ cpu ไม่นานนักและในที่สุดทุก process จะกลับเข้าไปรอใน I/O device queue ทำให้เกิดการว่างงานของ cpu (cpu idle) จนกระทั่ง cpu bounded process พร้องที่จะเข้าใช้ cpu ต่อก็จะเกิด cycle เดิม นั้นคือ process อื่นต้องรอใน ready queue (เมื่อ I/O operation complete) อีกเช่นเคยเหตุการณ์นี้ก่อให้เกิด Low CPU Utilization FCFS เป็นการ implement ในรูปแบบของ nonpreemptive scheme และไม่เหมาะสมใน time-sharing system

2.4.2. Shortest-Job-First Scheduling (SJF)

วิธีการของ Algorithm นี้คือการจัดสรร cpu ให้แก่ process ที่ต้องการใช้ cpu ที่มี cycle ของการใช้ cpu สิ้นสุดก่อน (หรือการจัด process ที่ใช้ cpu น้อยๆใน cycle ถัดไปให้เข้าใช้ cpu ก่อน process ที่มี cpu burst cycle ยาว)

ตัวอย่าง ถ้า $P_1 P_2 P_3$ และ P_4 มี cpu burst time ดังตาราง

Process	CPU Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

ถ้าใช้ SJF Algorithm ใน ready queue จะได้ลำดับของ process คือ

เวลา 0 3 9 16 24

Waiting Time	Milliseconds
P_4	0
P_1	3
P_3	9
P_2	16

Average waiting time = (0+3+9-16) / 4 = 7 milliseconds

ถ้าใช้ FCFS จะได้

Average waiting time = (0+6+14+21) / 4 = 10.25 milliseconds

ปัญหา ใน short-term scheduling จะไม่สามารถทราบ next cpu burst time ที่แน่นอนได้ นอกจากการประมาณค่า เช่น ประมาณค่าจาก cpu burst cycle ก่อน (ใน long-term scheduling อาจจะได้ค่าของ cpu burst time จากการกำหนดของ user ในช่วงของการ submit โปรแกรมนั้นๆเข้าสู่ ready queue)

SJF Algorithm เป็นได้ทั้ง preemptive และ nonpreemptive scheme ในกรณีที่ใช้รูปแบบของ preemtive scheme นั้นเมื่อมี job ใหม่เข้ามาสู่ ready queue ที่มี cpu burst time น้อยกว่า cpu burst time ส่วนของ process ที่กำลังใช้ cpu อยู่ process นั้นจะถูกขัดจังหวะ

2.4.3. **การจัดเวลาแบบวนรอบ** (RR : Round-Robin Scheduling)

เป็นการจัดสรรอีกรูปแบบหนึ่งที่ถูกออกแบบมาสำหรับระบบ time sharing โดยที่การทำงาน คล้ายกับ FCFS Scheduling นั่นคือ process ได้รับการจัดลำดับ ใน ready queue ตามลำดับการเข้าสู่ระบบ (เข้าสู่ queue) แต่อย่างไรก็ตาม process ที่กำลังใช้ CPU สามารถใช้ CPU ได้ในระยะเวลาจำกัดในแต่ละครั้ง นั่นคือมีการกำหนดช่วงเวลา ที่แต่ละ process จะใช้ CPU ได้ เรียกว่า time quantum หรือ time slice เมื่อ process ใด ๆ ใช้ CPU ครบเวลา 1 time quantum แล้ว process นั้นจะต้องออกจาก CPU กลับไปสู่ ready queue ในลำดับถัดไป และ CPU จะได้รับการจัดสรร ให้แก่ process ในลำดับต้น queue ใน ready queue ต่อไป

อย่างไรก็ตาม process สามารถออกจาก CPU ได้ก่อน เวลาที่กำหนด(1 time slice) ถ้า CPU Burst Cycle ของ process นั้นน้อยกว่า ค่าของ 1 time slice ที่กำหนดไว้

ถ้าพิจารณาจาก process ที่เข้าใช้ CPU ดังนี้

Process	Burst time
P1	24
P2	3
P3	3

ถึงแม้ว่าการจัดการในรูปแบบนี้ จะทำให้ process ต่างๆใน ready queue ได้รับการจัดสรรให้ใช้ CPU ในอัตราส่วนที่ใกล้เคียงกัน นั่นคือ แต่ละ process มีโอกาสได้เข้าใช้ CPU แน่นอน เมื่อเวลาผ่านไประยะหนึ่ง แต่ค่าเฉลี่ยในการรอคอยเข้าใช้ CPU ของการจัดในรูปแบบนี้จะค่อนข้างสูง โดยพิจารณาดังตัวอย่างคือ

ถ้าให้กลุ่มของ process ที่เข้าสู่ ready queue ตามลำดับคือ

Process	Burst time
P1	24
P2	3
P3	3

และให้ quantum time =4 milliseconds

(preempt) และนำออกจาก cpu เพื่อให้ job ที่มี cpu burst time น้อยกว่าเข้าใช้ cpu แทนดังนั้น ในบางครั้งวิธีการนี้จะเรียกว่า Shortest-Remaining-Time-First

หากวิธีการ implement ของ SJF เลือกใช้รูปแบบของ nonpreemtive scheme job ที่เพิ่งเข้ามาสู่ ready queue ต้องรอจนกว่า process ที่กำลังรอใช้ cpu ปล่อย cpu ใน cycle นั้นก่อนจึงจะได้รับการจัดสรร cpu ถึงแม้ว่า cpu burst time จะน้อยกว่า process ที่กำลังใช้ cpu ก็ตาม

ตัวอย่าง ถ้า implement แบบ preemptive scheme

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

ถ้า P_1 เข้าสู่ระบบก่อน (time 0)

ดังนั้น P_1 จะได้รับการจัดสรรให้ใช้ cpu ก่อน

เวลา 0

ต่อมา 1 milliseconds P_2 เข้าสู่ ready queue เนื่องจาก P_2 มี cpu burst time = 4 และน้อยกว่าเวลาที่เหลือของ P_1 (คือ 7 millisecond) ดังนั้น P_2 เข้าใช้แทนและ P_1 ต้องได้รับการจัดสรรเข้าสู่ cpu ใหม่ในรอบต่อๆไป

จะได้รูปแบบการรอคือ

เวลา 0 1 5 10 17 26

ดังนั้น average waiting time

P_1 P_2 P_3 P_4

= ((10-1) + (1-1) + (17-2) + (5-3)) / 4

เวลาที่เข้ามา

= 10.5 millisecond

ถ้าการจัดเป็นแบบ non-preemtive

Average waiting time = (0+8+12+17) / 4

= 7.75

จากลำดับการเข้าใช้ CPU จะเห็นได้ว่า P1 ได้รับการจัดสรรให้ใช้ CPU เป็น process แรก ในระยะเวลา 4 milliseconds แต่เนื่องจาก P1 ต้องการอีกทั้งหมด 20 milliseconds (Burst time=24) จึงต้องออกจาก CPU (โดยถูก preempt โดย CPU Scheduler) เพื่อให้ process ถัดไปใน ready queue (ในที่นี้คือ P2) เข้าใช้ CPU ต่อ แต่เนื่องจาก P2 ต้องการใช้ CPU เพียงแค่ 3 milliseconds ดังนั้น process นี้จึงปล่อย CPU ให้แก่ P3 ก่อนหมดเวลาใน 1 quantum time ในทำนองเดียวกัน เมื่อ P3 เข้าใช้ CPU จะออกจาก CPU ก่อนหมดเวลา ของ 1 quantum time หลังจากนั้น P1 สามารถกลับเข้าใช้ CPU ต่อ ลำดับการใช้ CPU (หรือการรอใน ready queue) สามารถเขียนในรูป gantt chart ได้คือ

0 4 7 10 14 18 22 26 30

ดังนั้นค่าเฉลี่ยของการรอคือ 17/3 = 5.66 milliseconds

(คือเวลา (0+7+10)/3)

P2 P3

RR Scheduling เป็นการจัดการแบบ Preemptive เสมอ เนื่องจากแต่ละ process สามารถเข้าใช้ CPU ได้มากที่สุดหรือนานที่สุด เท่ากับระยะเวลาของ 1 quantum time เท่านั้น ในการเข้าใช้ CPU หนึ่งรอบ และ process ที่ถูกออกจาก CPU และกลับไปรอใน ready queue burst time มากกว่า 1 quantum time

ในการจัดการแบบนี้ สามารถประมาณเวลาที่แต่ละ process ต้องรอใน ready queue ได้เสมอ นั้นคือ

ถ้าให้ มี process ทั้งหมดใน ready queue คือ n process

quantum time คือ q

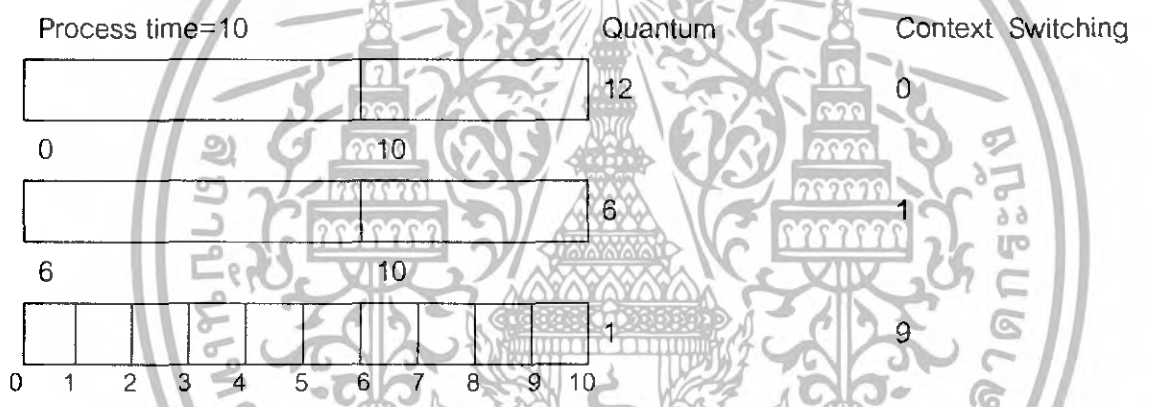
นั้นคือแต่ละ process จะได้รับการจัดสรรให้ใช้ CPU ทุกๆ 1/n (รอบละ q milliseconds) แต่ละ process จะรอนานที่สุดไม่เกิน (n-1)*q milliseconds ในแต่ละรอบ

ตัวอย่าง ถ้าให้มี process ใน ready queue ทั้งหมด 5 process และ quantum time คือ 20 milliseconds

จะได้ว่า แต่ละ process จะได้รับการจัดสรร CPU สูงสุด 20 milliseconds ในทุกๆ 100 milliseconds ในแต่ละรอบ (และ process จะรอไม่เกิน 80 milliseconds เพื่อเข้าใช้ CPU ในแต่ละครั้ง)

ประสิทธิภาพของการจัดสรร Scheduling แบบนี้ขึ้นอยู่กับขนาดของ quantum time เป็นหลัก ในกรณีที่ให้ quantum time กว้างมากเช่นมากกว่า CPU Burst time ของแต่ละ process จะได้ว่าการจัด CPU Scheduling ในกรณีนี้ก็คือ การจัดแบบ FCFS นั้นเอง แต่ถ้าให้ขนาดของ quantum time เล็กมาก (เช่น 1 milliseconds) แต่ละ process จะได้ รับการจัดสรร CPU ในระยะเวลาสั้นมากเช่นเดียวกัน ทำให้ user มีความรู้สึกว่า CPU ของตนเองโดยเฉพาะ (เนื่องจากได้รับการจัดสรร CPU ในช่วงระยะเวลาถี่มาก) และการจัดแบบนี้เรียกว่า process sharing

อย่างไรก็ตาม ในการกำหนดขนาดของ quantum time ใน RR Scheduling ต้องคำนึง overhead ที่จะเกิดจากการทำ Context Switching ด้วยเพื่อพิจารณาประสิทธิภาพโดยรวมของระบบ เนื่องจากถ้ากำหนด quantum time ขนาดเล็กจะก่อให้เกิด จำนวนครั้งของการเกิด Context Switching สูงมากกว่าในการกำหนดให้ ขนาดของ quantum time ใหญ่โดย พิจารณาจากตัวอย่างดังนี้



(ภาพนี้แสดงให้เห็นว่า จำนวน Quantum time ขนาดเล็กจะทำให้เกิดจำนวนครั้งของการเกิด Context Switching สูง)

นอกจากนี้ turn around time ของ process ก็ขึ้นอยู่กับขนาดของ Quantum time เช่นเดียวกัน ถ้าขนาดของ Quantum time เล็กมาก จำนวนของ Context Switching ที่เกิดขึ้นมากครั้ง จะเป็นตัวทำให้ turn around time ของ process เพิ่มขึ้นด้วยเช่นกัน และโดยปกติ turn around time ของ process ต่างๆ จะดีถ้า process สามารถ ทำงานได้เสร็จในช่วงของการได้รับการจัดสรร CPU ในรอบถัดไป (นั้นคือ burst time ที่เหลือจะน้อยกว่าขนาดของ quantum time จะไม่ต้องรอ CPU ในรอบถัดไปอีก)

2.4.4. การจัดเวลาตามลำดับความสำคัญ (Priority Scheduling)

เป็นการจัดลำดับ process ที่จะเข้าใช้ CPU ตามลำดับสิทธิ์ของแต่ละ process ดังนั้นในการจัดการ ในรูปแบบนี้ ทุก process จะมีค่าที่กำหนดสิทธิ์ (priority) ของตนเอง และ process ที่

มีค่าสิทธิ์สูงสุด จะได้รับการจัดสรรค์ CPU ให้ใช้ในช่วงเวลานั้น ๆ และ process ที่มี priority รองลงมา จะได้รับการจัดสรรค์ CPU ให้ใช้ถัดไปตามลำดับ

หมายเหตุ ค่าตัวเลขที่กำหนดค่า priority ไม่มีข้อจำกัดว่า ค่าตัวเลขน้อยหรือมาก(สูงหรือต่ำ) จะเป็นตัวเลขกำหนดค่า priority สูงสุดขึ้นอยู่กับ implementation ของแต่ละระบบ ในที่นี้จะใช้คำว่าลำดับสิทธิ์สูงสุด และลำดับสิทธิ์ต่ำ (high-priority / low-priority) สำหรับการจัดสรร CPU ให้แก่ process ในวิธีการนี้ ถ้าพิจารณาจากกลุ่มของ process ต่างๆ คือ P1,P2,..,P5 ซึ่ง เข้าขอใช้ CPU เมื่อเวลา time 0 โดยมี CPU -Burst ดังนี้

Process	Burst-time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

ถ้าใช้รูปแบบของ Priority Scheduling จะได้ลำดับของ process ที่พร้อมที่จะได้รับการจัดสรร CPU คือ

เวลา 0 1 6 16 18 19

ค่า priority กำหนดได้ทั้งจากภายใน (internallity defined) และจากภายนอก (externally defined) โดยที่การกำหนดจากภายในของระบบเอง สามารถทำได้โดยการ คำนวณจากค่าตัวแปรหรือค่าที่เกี่ยวข้องต่างๆ เช่น เวลา หรือ จำนวน memory ที่ process ต้องการ หรือแม้แต่จำนวนไฟล์ ที่ process ต้องการ ฯลฯ จะเห็นได้ว่าค่าเหล่านี้ขึ้นอยู่กับรูปแบบ และการทำงานของแต่ละ process และสามารคำนวณได้เมื่อ process นั้นๆ ใช้ทรัพยากรต่างๆ ของระบบ แต่ค่าลำดับสิทธิ์ ที่กำหนดจากภายนอก โดยทั่วไปจะเป็นการกำหนดจากขอบข่ายอื่นๆ ที่ไม่เกี่ยวข้องกับ การใช้ทรัพยากรของระบบโดยตรง เช่น การกำหนดความสำคัญของหน่วยงาน (เจ้าของ process) หรือค่าเช่าใช้ CPU ฯลฯ

ในการ implement การจัดรูปแบบนี้สามารถ เป็นไปได้ทั้งรูปแบบของ preemptive และ nonpreemptive นั้นคือ เนื่องจาก process ที่เข้าสู่ ready queue ตามค่า priority เสมอ ในกรณีที่ process ที่เข้าสู่ ready queue ใหม่ มีค่า priority สูงกว่า process ที่กำลังใช้ CPU จะได้รับการจัดสรรในรูปแบบใดรูปแบบหนึ่งดังนี้

- ได้รับการจัดสรรให้เข้าใช้ CPU ได้เลยทันที (process ที่กำลังใช้ CPU ต้องถูกนำออกจาก CPU ไปรอใน ready queue) → ถ้าเป็นการ implement แบบ preemptive scheme
- ได้รับการจัดสรรเป็นลำดับแรก สำหรับ process ที่รอเข้าใช้ CPU (ใน ready queue) → ถ้าเป็นการ implement แบบ nonpreemptive scheme

ปัญหาสำคัญที่เกิดขึ้นเนื่องจากการใช้ algorithm นี้คือ การที่ process ต่างๆ ซึ่งมี priority ต่ำหรือต่ำมาก ไม่ได้รับการจัดสรร CPU เลย ถ้าหากในระบบนั้นๆ มี process ที่มีค่า priority สูง เข้ามาสู่ระบบตลอดเวลา เนื่องจาก process เหล่านี้จะได้รับการจัดสรร CPU เสมอ และ process ที่มี priority ต่ำ จะต้องรอจนกว่าไม่มี process อื่น ๆ ซึ่งมี priority สูงกว่าอีกแล้ว ปัญหาในรูปแบบนี้เรียกว่า "Indefinite blocking" หรือ "starvation"

เพื่อแก้ปัญหาดังกล่าวได้นำวิธีการที่เรียกว่า "Aging" มาช่วย นั่นคือ การกำหนดให้ ค่า priority สูงขึ้น เมื่อ process นั้นๆ ต้องรอ ใน ready queue เป็นเวลานาน เพื่อสามารถเข้าแข่งขัน เพื่อขอใช้ CPU ได้ด้วย วิธีการนี้จะทำให้ process ที่มีค่า priority ต่ำ สามารถเข้าใช้ CPU ได้
เมื่อเวลาผ่านไป (ไม่ต้องรอแบบไม่มีกำหนด)

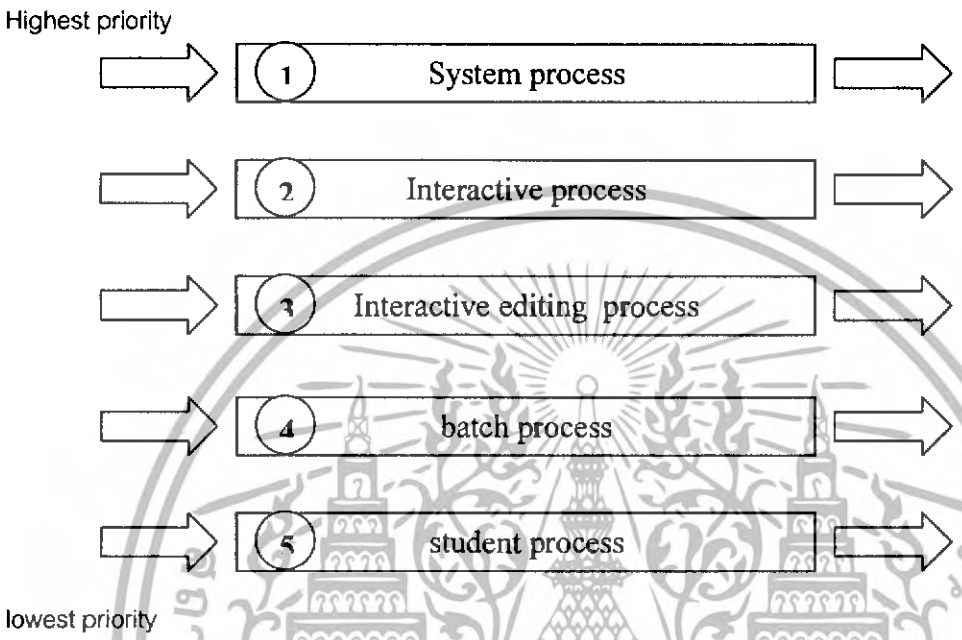
2.4.5. **การจัดเวลาแบบคิวหลายระดับ (Multilevel Queue Scheduling)**

เป็นการจัดรูปแบบการจัดสรร CPU ให้แก่ process อีกรูปแบบหนึ่ง และเหมาะสมในสิ่งแวดล้อม (ในระบบ) ที่การทำงานของ process ต่างๆ สามารถแบ่งออกได้เป็นหลายๆ ประเภท หรือหลายๆ รูปแบบที่ชัดเจน เช่นการทำงานในรูปแบบของ foreground (interactive) และ background (batch) process ซึ่งการทำงานของ process ในรูปแบบนี้จะแตกต่าง กันอย่างเห็นได้ชัดเจน เช่น process ในชนิดของ interactive ต้องการ response time ที่ดี (เวลาตอบสนองเร็ว) ในขณะที่ Batch process ต้องการให้ค่า turn around time น้อย เหล่านี้เป็นต้น

การจัดการแบบ Multilevel Queue Scheduling นี้ทำโดยการแบ่ง process ต่างๆ ออกเป็นกลุ่มๆ (กลุ่มละ 1 ready queue) แต่ละ process จะได้รับการจัดให้อยู่ในกลุ่ม เดียวตลอด โดยอาจจะแบ่งตามคุณลักษณะของ process นั้นๆ เช่น process priority หรือขนาดของ memory ที่ต้องการ ฯลฯ ในแต่ละ ready queue จะมีการจัดการของ process scheduling ที่อาจแตกต่างกันไป ตามความเหมาะสม เช่น ready queue ต่างๆ อาจจะแบ่งตามชนิดของ process เช่น background และ foreground process และในกลุ่มของ background process ใช้การจัดการแบบ FCFS ในขณะที่กลุ่มของ foreground process อาจจะใช้การจัดสรร CPU แบบ RR

นอกจากการจัดสรร CPU ในแต่ละ queue แล้ว ยังต้องมีการจัดลำดับ หรือ priority ระหว่าง queue ต่างๆ ด้วย เช่นกัน โดยทั่วไปจะจัดโดยวิธีการของ priority และ preemptive ดังตัวอย่างคือ

ถ้าให้การจัดสรร CPU แบบ Multilevel Queue Scheduling ประกอบด้วย process ต่างๆ 5 กลุ่ม ตามรูป



ในการจัดการแบบนี้ process ที่อยู่ใน low - priority queue จะต้องรอจนกว่า process อื่นๆ ในลำดับที่สูงกว่า เสร็จงาน จึงจะได้รับการจัดสรร CPU ให้ และในขณะที่ process ใน low - priority queue กำลังทำงาน และมี process เข้ามา ใน queue ที่ลำดับสูงกว่า process นั้น ๆ จะต้องถูกออกจาก CPU เพื่อให้ process ใหม่ ที่มี priority สูงกว่า ได้ใช้ CPU ก่อนเสมอ

นอกจากนี้ อาจจะทำการแบ่งสรรเวลา ในการใช้ CPU ให้แก่ queue ในระดับต่างๆ กัน (time slice) เช่น ให้ใช้ CPU ได้ 80 % ในบางกลุ่ม และอีก 20 % สำหรับอีกกลุ่ม เป็นต้น และในแต่ละกลุ่ม จะต้องบริหารจัดการกับเวลาที่ได้รับมา สำหรับ process ในกลุ่มเอง โดยวิธีการจัดการในรูปแบบที่เหมาะสมเอง

2.5. Shell Script

2.5.1. ลักษณะทั่วไปของ Shell

หน้าที่พื้นฐานของ Shell โดยทั่วไปนั้นคือการรับคำสั่งจาก User เพื่อทำการบอก OS ให้ Load และ Run Program ต่างๆ อย่างไรก็ตามได้มีการพัฒนาให้ Shell มีความสามารถในการประมวลผลคำสั่งต่างๆ เพื่อให้ทำงานในลักษณะของ Batch ได้ ทั้งนี้โครงสร้างของโปรแกรมต่างๆ หรือ Shell นั้นจะประกอบด้วยตัวแปรระบบ หรือ สภาวะแวดล้อม(Envirenment) ซึ่งพร้อมที่จะส่ง

ให้โปรแกรมอื่นๆ ประมวลผลต่อ ประโยคเงื่อนไขต่างๆ ระบบ Profile สำหรับวัดเวลาในการประมวลผลของโปรแกรมต่างๆ และ User Defined Function

ในการวัดเวลาที่ใช้ในการทำงานคำสั่งต่างๆ ของ Unix นั้นสามารถทำได้โดยใส่ time นำหน้าคำสั่งที่ต้องการประมวลผล ซึ่งโปรแกรมจะบอก OS ให้ทำการจับเวลา และ รายงานผลเวลาที่ใช้ในการประมวลผลหลังจากทำงานเสร็จ เช่น

```
time ls -al /bin/*
```

สำหรับกรณี Shell บน Unix นั้น การเขียนคำสั่งแต่ละคำสั่งจะต้องจบคำสั่ง ด้วยบรรทัดใหม่ หรือ เครื่องหมาย semi-colon เสมอ ";" และกรณีที่คำสั่งมีความยาวมาก (ไม่จบภายใน 1 บรรทัด) ให้จบบรรทัดด้วยเครื่องหมาย "" แล้วพิมพ์ต่อในบรรทัดถัดไป เช่น

```
echo "Test"; ls;

echo "Hello world"
```

การเขียน Comment ใน Shell Script จะใช้เครื่องหมาย "#"

ทั้งนี้ในการเขียนโปรแกรมที่เป็นภาษา Script ต่างๆ บน Unix นั้น ในบรรทัดแรกเราควรจะระบุชื่อโปรแกรมที่จะทำการประมวลผล Script นั้นๆ ไว้ด้วยเช่น

```
#! /bin/ksh
```

หรือ กรณีเป็น Perl อาจจะเขียนเป็น

```
#! /usr/bin/perl
```

เป็นต้น

2.5.2. การใช้งาน Variable / Environment

บนระบบ OS โดยทั่วไปมักจะมี Environment หรือ System Variable เพื่อส่งผ่านค่าระหว่างโปรแกรมต่างๆ ภายในระบบ เสมอ ซึ่งโดยทั่วไปเราจะอ่านค่า Environment (ของ OS เกือบทุกตัว) ได้โดยการใช้คำสั่ง set

ตัวอย่างการตั้งค่า Variable ใน Shell เช่น

bash / ksh	tcsh
myname="Krerk Piromsopa" count=5;	set myname="Krerk Piromsopa" set count=5;

นอกจากนี้เรายังสามารถนำผลลัพธ์จากการ Run Programm อื่นๆ มาเป็นค่า Variable ได้อีกด้วย โดยใช้ Back Quote แทน หรือ Run คำสั่งภายในเครื่องหมาย $(command) เช่น

bash / ksh	tcsh

list = `ls`
# หรือ 	
list=$(ls)
	set list = `ls`
	# หรือ
	set list=$(ls)

ทั้งนี้หากต้องการให้ค่า Variable หรือ Environment คงอยู่หลังจากที่ Run Shell สามารถทำได้โดยการ export เช่น

export myname,count

นอกจากนี้ยังมีตัวแปรระบบที่จะใช้สำหรับการอ้างอิงในโปรแกรม Shell ได้อีกด้วย เช่น

ตัวแปร	ความหมาย
$#	แสดงจำนวน Parameter ที่ส่งมาให้กับระบบ
$?	เก็บค่าที่ได้จากประมวลผลคำสั่งก่อนหน้านี้
$0	แสดงชื่อของ โปรแกรมที่กำลังประมวลผลอยู่
$*	เก็บค่า Parameter ทุกตัวที่ใช้ในการประมวลผล ($1 $2 ....)
"$@"	เก็บค่า Parameter ทุกตัวที่ใช้ในการประมวลผล ("$1" "$2" ....)

การใช้งาน Variable ทุกตัวนั้นจะเป็น Implicit Declaration กล่างคือผู้ใช้ไม่จำเป็นต้องประกาศตัวแปร เช่นในภาษา C หรือ ภาษาชั้นสูงอื่นๆ และ เมื่อต้องการยกเลิกตัวแปรนั้นสามารถทำได้โดยใช้คำสั่ง unset varname เช่น

unset myname

test และ expression

บน bash และ ksh จะมีคำสั่ง Test และการเขียนเงื่อนไขในการตรวจสอบ Expression ต่างๆ ในการเขียนประโยคเงื่อนไขต่าง ซึ่งมีโครงสร้างการเขียนดังนี้

test *expression*

หรือ

[*expression*]

สำหรับ tcsh นั้นจะไม่มีคำสั้ง test แต่จะมีการนำ expression เพื่อไปใช้ในการประมวผลผลคำสั่งต่างๆ เช่นกัน

ทั้งนี้ expression บน bash กับ ksh และ tcsh นั้นจะมีโครงสร้างที่แตกต่างกัน ดังอธิบายได้ดังต่อไปนี้

bash / ksh	tcsh	ความหมาย
คำสั่งเกี่ยวกับจำนวนเต็ม		
int1 -eq int2	int1 == int2	เป็นจริงเมื่อ int1 เท่ากับ int2
int1 -ge int2	int1 >= int2	เป็นจริงเมื่อ int1 มากกว่าหรือเท่ากับ int2
int1 -gt int2	int1 > int2	เป็นจริงเมื่อ int1 มากกว่า int2
int1 -le int2	int1 <= int2	เป็นจริงเมื่อ int1 น้อยกว่าหรือเท่ากับ int2
int1 -lt int2	int1 < int2	เป็นจริงเมื่อ int1 น้อยกว่า int2
int1 -ne int2	int1 != int2	เป็นจริงเมื่อ int1 ไม่เท่ากับint2
คำสั่งเกี่ยวกับ String		
str1 = str2	str1 == str2	เป็นจริงเมื่อ str1 เหมือนกับ str2
str1 != str2	str1 != str2	เป็นจริงเมื่อ str1 ไม่เหมือนกับ str2
str		เป็นจริงเมื่อ str ไม่เป็น null
-n str		เป็นจริงเมื่อ str มีความยาวมากกว่า 0
-z str		เป็นจริงเมื่อ str มีความยาวเป็น 0
คำสั่งเกี่ยวกับ file		
-d filename	-d filename	เป็นจริงเมื่อ filename เป็น directory
-f filename	-f filename	เป็นจริงเมื่อ filename เป็น file
-r filename	-r filename	เป็นจริงเมื่อ filename อ่านได้โดยโปรแกรม

-s filename		เป็นจริงเมื่อ filename มีขนาดไม่เป็น 0
-w filename	-w filename	เป็นจริงเมื่อ filename เขียนได้โดยโปรแกรม
-x filename	-x filename	เป็นจริงเมื่อ filename run ได้โดยโปรแกรม
	-e filename	เป็นจริงเมื่อ filename มีชื่ออยู่
	-o filename	เป็นจริงเมื่อ filename เป็นของผู้ใช้ปัจจุบัน
	-z filename	เป็นจริงเมื่อ filename มีขนาดเป็น 0
คำสั่งเกี่ยวกับ Logical อื่นๆ		
! expr	! expr	เป็นจริงเมื่อ exp เป็นเท็จ
exp1 -a exp2	exp1 && exp2	เป็นจริงเมื่อ exp1 และ exp2 เป็นจริง
exp1 -o exp2	exp1 || exp2	เป็นจริงเมื่อ exp1 หรือ exp2 เป็นจริง

2.5.3. เงื่อนไข if

โครงสร้าง

bash / ksh	tcsh

if [expression]
then
  commands
elif [exprression]
then
  commands
else
  commands
fi
	if (expression) then
	  commands
	else if (expression) then
	  commands
	else
	  commands
	endif

ทั้งนี้จะมี elif หรือ ไม่มี else ก็ได้

ตัวอย่างการใช้งาน เช่น

```
#! /bin/ksh
if [ -f .signature ]
then
    echo "There is .signature file in this current directory"
else
    echo "File .signature could not be found.
fi
```

2.5.4. เงื่อนไข while

โครงสร้าง

bash / ksh	tcsh
while [expression]	while (expression)
do	commands
commands	end
done	

ตัวอย่างการใช้ while เช่น

```
#! /bin/ksh
count=1
while [ -n "$*" ]
do
echo "This is parameter no $count $1"
shift
count=`expr $count+1`
done
```

2.5.5. คำสั่ง echo

ใช้เพื่อการแสดงข้อความ

2.5.6. การประมวลผลทางคณิตศาสตร์ และ expression

2.5.6. การประมวลผลทางคณิตศาสตร์ และ expression

เราสามารถประมวลผลบวกลบคูณหาร หรือ expression ต่างๆ ได้โดยใช้โปรแกรม expr ช่วย เช่น

a=`expr $a + 1` หรือ set a=`expr $a + 1`

ในกรณีของ bash เราสามารถใช้ a=$(($a + 1)) แทนได้เลย

ข้อควรระวังคือ จะต้องมีการเว้นวรรคระหว่าง ตัวแปร ค่าคงที่ และ เครื่องหมาย (มิเช่นนั้นจะได้ผลลัพธ์เป็น string ต่อกัน) เครื่องหมายและการปฏิบัติการที่ใช้ได้ใน expr คือ +, -, *, / , % (modulo หรือ หารเอาเศษ) นอกจากนี้ยังมีการเปรียบเทียบต่างๆ เช่น =, <, >, <=, >=, และ != (กรณีใช้กับ bash $((expression)) ให้นำเครื่องหมาย ออก การเปรียบเทียบเหล่านี้จะให้ผลลัพธ์เป็น 0 หากเป็นเท็จ และ 1 หากเป็นจริง

2.6. XML (Extensible Markup Language)

ทุกวันนี้ ภาษา XML เป็นแนวทางหนึ่งที่จะสร้างคุณค่าให้กับผู้ใช้อินเทอร์เน็ต และเป็นหนทางที่จะสร้างบทบาทการประยุกต์ใช้งานบนเครือข่าย และเป็นภาษาที่กำลังมาแรงมากที่สุด เพราะกำลังกลายมาเป็นภาษามาตรฐานสำหรับการพัฒนาระบบการแลกเปลี่ยนเอกสาร และข้อมูลอิเล็กทรอนิกส์ ซึ่งกำลังจะนำศักยภาพใหม่ของโทคโนโลยีที่มีการพัฒนาอย่างไม่หยุดยั้งเพื่อเข้ามารองรับในการประมวลผลบนอินเทอร์เน็ต โดยเป็นเทคโนโลยีระบบเปิดที่เน้นการสื่อสารระหว่างผู้คนบนอินเทอร์เน็ตและแอพพลิเคชัน ซึ่งมีความสามารถทำงานได้ระหว่างแพลตฟอร์มที่แตกต่างกัน

XML (Extensible Markup Language) ถูกออกแบบมาเพื่อให้ผู้สร้างเอกสารสามารถนำไปใช้งานในรูปแบบวิธีการที่ง่าย มีความชัดเจนและเป็นเซตย่อยของ SGML (Standard Generalized Markup Language) ซึ่งเป็นภาษาที่นิยมใช้และได้รับการพัฒนาให้มีประสิทธิภาพสูงสุดในการทำงานบนเว็บ โดย XML จะประกอบด้วย 3 ส่วนพื้นฐานด้วยกัน คือ เอกสารข้อมูล (Data document) เอกสารนิยามความหมาย (definition document) และนิยามภาษา (definition language)

ปัจจุบัน มี 2 ภาษาด้วยกันที่มีการสร้างนิยามภาษาของเอกสารข้อมูลภาษาอื่นได้ คือ SGML และ XML เช่น ภาษา WML(Wireless Markup Language) ก็มีต้นกำเนิดมาจาก XML ที่ใช้ในการแสดงข้อความบนโทรศัพท์มือถือระบบ WAP (Wireless Application Protocol) โดยที่ XML ถือได้ว่าเป็นส่วนหนึ่งของ SGML ที่เป็นข้อกำหนดในการสร้างหรือจัดทำเอกสารในรูปแบบ

อิเล็กทรอนิกส์ที่กำหนดโดย W3C หรือ World Wide Web Consortium ใน SGML มีกระบวนการเรียกว่า Information Analysis สำหรับใช้ตรวจสอบโครงสร้างและราย ละเอียดของข้อมูล กระบวนการดังกล่าวเรียกว่า DTD (document type definition) ซึ่งเป็นตัวชี้เนื้อหาของ ออบเจ็กต์ (Object) ในกลุ่มข้อมูล

XML เป็นภาษาที่มีลักษณะเป็น tag คล้าย HTML แต่ไม่ได้มุ่งที่การแสดงผล XML มุ่งที่การสื่อความหมายโดยอนุญาตให้ผู้ใช้สามารถกำหนด tag ขึ้นได้เองเพื่อให้สื่อความหมายทางภาษาของมนุษย์แต่คอมพิวเตอร์เองก็เข้าใจเช่นกัน ทำให้ข้อมูลระหว่าง tag สามารถนำไปประมวลผลต่อได้เช่น

```
<ComputerBook>
<book>
<name>เว็บเซอร วิส</name>
<price>10.00$</price>
</book>
<book>
<name>xml</name>
<price>10.00$</price>
</book>
</ComputerBook>
```

2.6.1. คุณลักษณะต่างๆของ XML

XML สามารถที่จะจัดการได้หลายรูปแบบทั้งองค์ประกอบ โครงสร้างเอกสาร ลักษณะประเภท แอตทริบิวต์ และอิลิเมนต์ โดยเป็นภาษาที่ถูกออกแบบมาเฉพาะสำหรับการพัฒนาโปรแกรมเว็บเพื่อการจัดส่งข้อมูลสารสนเทศ ตลอดจนถูกนำมาใช้สร้างภาษามาร์คอัพ (markup) นั้นเอง ซึ่งตรงกันข้ามกับ SGML ที่มีความซับซ้อนมากกว่า ส่วน HTML ก็เป็นเอกสารไม่มีความยืดหยุ่นต่อการใช้งาน อย่างไรก็ตามในทางปฏิบัติเอกสาร XML มีกฎพื้นฐานเพื่อให้การสร้างเอกสารมีรูปแบบที่ถูกต้อง ในการใช้งานจริงโดยปกติแล้ว XML สามารถจัดเก็บฐานข้อมูล กำหนดโครงสร้างเอกสาร การนำเสนอมัลติมีเดียต่างๆ การจัดเก็บกราฟิกที่มีลักษณะแบบเวกเตอร์ ตลอดจนการสื่อสารระหว่างโปรแกรมต่างๆ และนอกจากนี้แล้ว XML ยังสามารถช่วยในการประมวลผลข้อมูลแล้วส่งผ่านให้โปรแกรมประยุกต์ไปยังแหล่งเก็บข้อมูล อย่างเช่น ข้อความหรือข้อมูล เป็นต้น

XML เป็นเอกสารที่เขียนด้วยข้อความปกติธรรมดา คุณจึงสามารถสร้างเอกสารหรือแก้ไขไฟล์ XML ได้อย่างง่ายดาย โดยการใช้โปรแกรมแก้ไขข้อความ (Text Editor) ซึ่งเป็นมากกว่าเครื่องมือการเขียนโปรแกรมที่ไม่ซับซ้อน หรือถ้าหากต้องการใช้โปรแกรมที่มีความสามารถพิเศษมากกว่านี้ ก็ต้องใช้โปรแกรมแก้ไขข้อความที่อยู่ในชุดโปรแกรม Microsoft Visual Studio เช่น Microsoft Visual C++ Microsoft Visual Basic และ Microsoft Visual Foxpro เป็นต้น ภาษา XML ใช้แท็กเริ่มต้นและแท็กปิดเสมอเช่นเดียวกับ HTML ซึ่งเรียกว่า อิลิเมนต์ ซึ่งเป็นการแบ่งแยกระหว่างข้อมูลและคำสั่ง เพื่อระบุว่าข้อมูลที่อยู่ระหว่างแท็กดังกล่าวคือ ข้อมูลอะไร

ส่วนประกอบในเอกสาร XML มีอยู่ 2 ส่วนหลักด้วยกัน คือ Prolog Element และ Document Element (หรือ Root Element) ในส่วนของเอกสาร XML คือ อิลิเมนต์ เดี่ยว ซึ่งสามารถบรรจุ อิลิเมนต์ เพิ่มเติมในเอกสาร XML ได้ โดยในเอกสาร XML นั้น อิลิเมนต์จะแสดงลักษณะโครงสร้างของเอกสาร และจะแสดงส่วนประกอบเนื้อหาของเอกสารอยู่ภายในสัญลักษณ์ อิลิเมนต์ ประกอบด้วยแท็กเริ่มต้น (start-tags) เนื้อหาภายใน อิลิเมนต์ และแท็กสิ้นสุด (end-tags) ส่วนเนื้อหาภายใน อิลิเมนต์ สามารถเป็นได้ทั้งข้อมูลหรือ อิลิเมนต์ อื่นๆ ที่ซ้อนอยู่ภายในหรือทั้งสองแบบ

2.6.2. ไวยากรณ์ และ คำสั่งพื้นฐาน

2.6.2.1. การเปิด-ปิดแท็ก XML

- กรณีที่มีข้อความภายใต้แท็ก รูปแบบการระบุแท็กก็คือ <tag>-----ข้อความ-----</tag>
- กรณีที่ไม่มีข้อความภายใต้แท็ก รูปแบบการระบุแท็กคือ <tag/> ถ้าภายในแท็กว่างมีการกำหนดแอตทริบิวต์ ก็ต้องลงท้ายด้วยเครื่องหมาย / เช่นกัน รูปแบบการระบุแท็ก คือ

<tag attribute="value"/>

2.6.2.2. อย่ากำหนดชื่อแท็กเหลื่อมกัน

ตัวอย่างที่ถูกต้อง <a> this is outer zone this is inner zone

ตัวอย่างที่ผิด <a> this is outer zone this is inner zone

2.6.2.3. Case sensitivity ของชื่อแท็ก

ภาษา XML ให้ความสำคัญกับอักษรตัวพิมพ์เล็ก – ตัวพิมพ์ใหญ่เป็นพิเศษฉะนั้นชื่อแท็กที่ตั้งขึ้นเองจึงต้องระวังไว้ด้วยว่า อักษรตัวพิมพ์เล็ก – ตัวพิมพ์ใหญ่แตกต่างกันตัวอย่างเช่น

<Tag>, <TAG>, <tag>, <tAG>

2.6.2.4. ค่าแอตทริบิวต์ต้องมีเครื่องหมายคำพูดกำกับไว้เสมอ

ขอยกตัวอย่างแท็กในภาษา HTML เช่นแท็กที่เกี่ยวกับไฟล์รูปภาพ ดังนี้

<img src="/pic/image1.gif" alt="bird">

แอตทริบิวต์ src ก็คือ ส่วนขยายเพื่อบอกตำแหน่งที่อยู่ของไฟล์รูปภาพโดย src ถือว่าเป็นชื่อแอตทริบิวต์ ส่วน /pic/image1.gif เรียกว่าเป็นค่าของแอตทริบิวต์

2.6.2.5. **อักษรพิเศษบางตัวต้องแทนด้วยรหัสพิเศษ**

เราไม่สามารถระบุอักขระพิเศษลงในเอกสาร XMLได้โดยตรง เพราะว่าเบราเซอร์จะตีความหมายว่าเป็นสัญลักษณ์การทำงาน ในกรณีที่จำเป็นต้องระบุสัญลักษณ์พิเศษลงไปเราต้องแทนด้วยรหัสพิเศษเท่านั้น เช่น

เครื่องหมาย < แทนด้วย <

เครื่องหมาย < แทนด้วย >

เป็นต้น

2.6.3. DTD **คืออะไร**

DTD คือแฟ้มข้อมูล (หรือหลายแฟ้มข้อมูลที่ใช้งานร่วมกัน) ซึ่งบรรจุข้อกำหนด และกฎเกณฑ์ของเอกสาร ชุดข้อกำหนดเหล่านี้ สำหรับการกำหนดรูปแบบอิลิเมนต์ ตัวอย่างเช่น หากต้องการเอกสารที่ม อิลิเมนต์ <LIST> ที่มีอิลิเมนต์ <ITEM> บรรจุอยู่ภายในข้อกำหนดในแฟ้มข้อมูล DTD
จะมีรูปแบบดังนี้

<!ELEMENT item (#pcdata)>

<!ELEMENT list (item)+>

ซึ่งอธิบายความหมายคืออิลิเมนต์ items บรรจุข้อความใดๆ และอิลิเมนต์ list บรรจุอิลิเมนต์ item อีกที ดังนั้น DTD เป็นรูปแบบภาษา ซึ่งทำให้สามารถตรวจสอบเอกสาร ที่นำเอาข้อกำหนด DTD ไปใช้ ว่าถูกจัดสร้างตามความต้องการหรือไม่ ทำให้ระบบการ rendering สามารถเข้าใจตัวเอกสารได้ดี และดึงไปใช้งานได้ถูกต้อง

2.6.4. XML Schema **คืออะไร**

ภาษา XML สกีมา หรือ XML Schema มีหน้าที่หลักคือ กำหนดโครงสร้างที่สมบูรณ์ให้กับเอกสาร XML กำหนดข้อบังคับที่เอกสาร XML ต้องปฏิบัติตาม กำหนดชนิดและคุณสมบัติของข้อมูลที่ใช้ในเอกสาร XML schemas ยังสนับสนุนรูปแบบในการทำงานของผู้พัฒนานิยามภาษา (Programming language) กำหนดคำจำกัดความให้กับเอกสารนิยามข้อมูล (Program) สำหรับเอกสารข้อมูล (Input/Output) เพื่อขยายขีดความสามารถให้กับภาษาข้อมูลให้เป็นลักษณะเฉพาะเจาะจง โดยกำหนดขอบเขตแอพพลิเคชันและการติดต่อข้อมูลข่าวสารกับระบบอื่นๆ

2.6.5. **ความแตกต่างระหว่าง DTD และ** XML Schema

ความแตกต่างที่สำคัญระหว่าง DTD และ XML schema คือ ความชัดเจนและความสามารถของทั้ง 2 ภาษา โดย Schema ได้จัดสร้างประโยชน์ต่างๆ มากมายพร้อมกับมีความสามารถโดดเด่นที่เหนือกว่า DTD

ภาษา XML สกีมา หรือ XML Schema มีหน้าที่หลักคือ กำหนดโครงสร้างที่สมบูรณ์ให้กับเอกสาร XML กำหนดข้อบังคับที่เอกสาร XML ต้องปฏิบัติตาม กำหนดชนิดและคุณสมบัติของข้อมูลที่ใช้ในเอกสาร XML ส่วน DTD นั้นโดยทั่วไปแล้วจะเป็นตัวกำหนดเค้าโครงของกฎทั้งหมดให้กับอิลิเมนต์แต่ละตัวตามแต่ละชนิดของเอกสาร ซึ่งใช้บ่งบอกถึงชนิดของข้อมูลที่ใช้งานอยู่ในเอกสารนั้นๆ กับเอกสาร XML แต่ DTD ก็ยังมีข้อด้อยเมื่อเทียบกับ XML สกีมา อย่างไรก็ตามทั้ง DTDs และ schemas ยังสนับสนุนรูปแบบในการทำงานของผู้พัฒนานิยามภาษา (Programming language) กำหนดคำจำกัดความให้กับเอกสารนิยามข้อมูล (Program) สำหรับเอกสารข้อมูล (Input/Output) เพื่อขยายขีดความสามารถให้กับภาษาข้อมูลให้เป็นลักษณะเฉพาะเจาะจง โดยกำหนดขอบเขตแอพพลิเคชันและการติดต่อข้อมูลข่าวสารกับระบบอื่นๆ

Schemas อนุญาตให้คุณกำหนด อิลิเมนต์ ที่มีได้หรือต้องการมีในเอกสาร อีกทั้งยังให้คุณจัดการแยกข้อมูลออกจากภาษา เช่น ในฐานข้อมูลเชิงวัตถุ (object-oriented) ยิ่งกว่านั้น Schemas ยังถูกออกแบบให้เป็นเครื่องมือในการจัดการรวบรวมหรือผสมผสานเอกสารหลายเอกสารเข้าด้วยกันได้อย่างง่าย นอกจากนี้ยังอนุญาตให้คุณอธิบายข้อจำกัดของข้อมูลเพื่อจะได้แสดงรูปแบบและขนาดของข้อความได้อย่างถูกต้องแม่นยำ ส่วนภาษา XML parsers ที่อยู่ในตลาดทั้งหมดทุกวันนี้ยังสนับสนุนรูปแบบการทำงานที่มีข้อมูลหลายชนิด(DTD-type) อย่างไรก็ตามในยุคที่สองจะใช้ XML schema แทน ซึ่งจะมีการอธิบายทั้งรูปแบบโครงสร้างของลำดับ (syntax) และชนิดของข้อมูลด้วย

คำแนะนำเกี่ยวกับโครงสร้างข้อมูลที่เป็นเอกสาร XML สามารถแสดงคำสั่งและถูกบรรจุใน 2 เทคโนโลยี คือ DTD ซึ่งกำหนดกฎเกณฑ์รูปแบบเอกสาร XML และส่วน XSL (eXensible Style Sheet Language) จะบอกไปยังคอมพิวเตอร์ที่ใช้ว่าจะมีวิธีการจัดรูปแบบหรือสร้างสไตล์ชีตเพื่อแปลงเอกสาร XML ทั้งหลายไปเป็นเอกสารชนิดอื่นได้อย่างไร DTD สามารถถูกบรรจุไว้ในเอกสาร XML และสามารถถูกเชื่อมต่อภายนอกตามต้องการ

จริงๆ แล้ว XSL เป็นภาษาที่ใช้สำหรับแปลงหรือจัดรูปแบบให้กับเอกสาร XML โดยสามารถนำข้อมูลออกจากเอกสาร XML ของผู้ใช้ จากแบบหนึ่งไปสู่อีกแบบ การแปลงเอกสารมีประโยชน์ในกรณีที่บริษัทแห่งหนึ่งใช้ Schema แบบหนึ่งและลูกค้าใช้ Schema อีกแบบหนึ่ง แต่บริษัทแห่งนี้สามารถแปลงเอกสาร XML ของลูกค้าที่ใช้ Schema ต่างกันมาเป็น Schema ของ

บริษัทได้ โดยข้อมูลภายในเอกสาร XML ยังเหมือนเดิม อีกทั้งสามารถปรับแต่งข้อมูลให้ตรงตามชนิดอุปกรณ์รอบข้างและยังสามารถเขียนสคริปต์เพื่อแปลงเอกสารได้ด้วย

เทคโนโลยีสำคัญที่ยังไม่ได้กล่าวถึงนั้นก็คือ SOAP (Simple Object Access Protocol) เป็น XML-based โปรโตคอล (lightweight protocol) สำหรับการแลกเปลี่ยนข้อมูลในสภาวะแวดล้อมแบบกระจายศูนย์ SOAP ได้กำหนดเมเสจจิ้งโปรโตคอล (Messaging Protocol) ระหว่างผู้ขอบริการ (requestor) กับผู้ให้บริการ (provider) จุดเด่นของ SOAP ก็คือเป็นโปรโตคอลที่เป็นกลางในการรับและส่งข้อมูล กล่าวคือ ไม่มีใครเป็นเจ้าของและเป็นโปรโตคอล ที่ทำงานกับโปรโตคอลอื่นหลายชนิด

2.6.6. XPath

XPath คือ ผลจากการพยายามที่จะจัดหาโครงสร้างประโยคทั่วๆไป และการศึกษาสัญลักษณ์สำหรับการใช้ร่วมกันระหว่าง XSLT และ XPointer ซึ่งจุดประสงค์หลักของ XPath คือการจัดการปัญหาในส่วนของเอกสาร XML ในการสนับสนุนจุดประสงค์นี้ก็ได้จัดเตรียมสิ่งอำนวยความสะดวกพื้นฐานสำหรับการจัดการสตริง ตัวเลข และ ตรรกะ XPath ใช้คำสั่งที่กระชับ ไม่ใช้โครงสร้างประโยคของ XML ในการอำนวยความสะดวกต่อการใช้ XPath ในค่าแอตริบิวท์ของ URI และ XML ซึ่ง XPath ปฏิบัติบนนามธรรม โครงสร้างทางตรรกะของเอกสาร XML ตรงข้ามกับโครงสร้างประโยคภายนอก

2.7 มัลติมีเดียร์ไฟล์

เนื่องจากโครงงานฉบับนี้ได้ทำการตัดไฟล์มัลติมีเดียร์ให้เป็นส่วนเล็กๆ เพื่อส่งกระจายไปยังเครื่องต่างๆ ในระบบกริดเพื่อทำการเข้ารหัสให้เป็น MPEG4 จึงจัดทำส่วนเนื้อหานี้ขึ้นเพื่อให้ทำความเข้าใจในเรื่องของมัลติมีเดียร์ไฟล์ได้ง่ายยิ่งขึ้น

สิ่งที่ต้องทำความรู้จักในเรื่องของมัลติมีเดียร์ไฟล์จะประกอบด้วย 2 ส่วนหลัก คือ Codec และ Container ซึ่งรายละเอียดมี ดังนี้

Codec เป็นส่วนประกอบที่เป็นอุปกรณ์หรือโปรแกรมที่แสดงถึงการเข้ารหัสและถอดรหัสบนสัญญาณหรือสายของข้อมูลดิจิตอล เป็นรูปแบบของข้อมูลภายใน โดยส่วนใหญ่แล้ว codec จะถูกใช้มากในแอพพลิเคชั่นที่ทำงานเกี่ยวกับวีดีโอคอนเฟอเรนซ์และสตรีมมิงมีเดียร์ และหลายประเภทที่ได้รับการออกแบบมาโดยให้ความสำคัญกับลักษณะของตัวมีเดียร์ที่จะถูกทำการเข้ารหัส ตัวอย่างเช่น ดิจิตอลวีดีโอ (ใช้ DV codec) ที่ใช้สำหรับการเก็บภาพเหตุการณ์ทางกีฬา เช่น เบสบอลหรือฟุตบอล ที่ต้องการใช้งานในเรื่องของการเคลื่อนไหวสูงแต่ไม่มีความจำเป็นมากในเรื่องของสีภาพที่ถูกต้อง ในขณะที่วีดีโอสำหรับงานมหกรรมการแสดงต่างๆนั้นจำเป็นในเรื่องการแสดงผลของสีสันและเท็กซ์เจอร์ของพื้นผิวมาก ส่วนในเรื่องของข้อมูลทางมัลติมีเดียร์นั้นมีความ

ต้องการทั้งในส่วนของ audio และ video ซึ่งต้องฝังอยู่ในรูปแบบของ Container ตัวอย่างของ codec ต่างๆ มีดังนี้

Audio codecs

- Apple Lossless
- Audio Lossless Coding (MPEG-4 ALS)
- Direct Stream Transfer (DST)
- Free Lossless Audio Codec (FLAC)
- LA (Lossless Audio)

Video codecs

- CorePNG
- H.264
- Huffyuv
- Lagarith
- MJPEG
- MPEG-1 Video
- MPEG-2 video
- MPEG-4 Advanced Simple Profile Video

Container format คือ ลักษณะของไฟล์คำนวณที่บรรจุรูปแบบของข้อมูลที่หลากหลายที่บีบอัดโดยมาตรฐานของ codec รูปแบบต่างๆ คอนเทนเนอร์นั้นใช้ในการแยกแยะและแทรกรูปแบบข้อมูลที่แตกต่างกันของ data types คอนเทนเนอร์โดยปกตินั้นบรรจุ audio codec ที่แตกต่างกันได้หลากหลาย ในขณะเดียวกันคอนเทนเนอร์ขั้นสูงบางชนิดจะสนับสนุนทั้ง audio, video, subtitles และอื่นๆอีกมาก ตัวอย่างของคอนเทนเนอร์ ดังนี้

- WAV (รูปแบบ RIFF file ใช้อย่างกว้างขวางในแพลตฟอร์มวินโดว์)
- AIFF (รูปแบบ IFF file ใช้อย่างกว้างขวางในแพลตฟอร์ม Mac OS)

คอนเทนเนอร์รูปแบบอื่นที่สามารถเก็บรูปแบบ audio และ video ได้หลายชนิดและเป็นที่นิยมใช้อย่างแพร่หลาย ดังนี้

- IFF (แพลตฟอร์มแรกของคอนเทนเนอร์)
- AVI (คอนเทนเนอร์มาตรฐานของวินโดว์ซึ่งมีพื้นฐานจาก RIFF)
- MOV (คอนเทนเนอร์มาตรฐานของ QuickTime)
- MPEG-2 transport stream (คอนเทนเนอร์มาตรฐานสำหรับการบรอดคาสท์ดิจิตอลซึ่งปกติจะบรรจุ video และ audio stream และโปรแกรมแนะนำระบบ)

- MP4 (คอนเทนเนอร์มาตรฐานสำหรับ MPEG-4)
- Ogg (คอนเทนเนอร์มาตรฐานสำหรับ Xiph.org codecs)
- ASF (คอนเทนเนอร์มาตรฐานสำหรับ WMA และ WMV)
- RealMedia (คอนเทนเนอร์มาตรฐานสำหรับ RealVideo และ RealAudio)
- Matroska (เป็น open standard)
- 3gp (ใช้ในโทรศัพท์มือถือ)

2.8 โปรแกรม Mencoder

Mencoder คือ เครื่องมือ หรือ tool ที่ใช้ในการ encoding decoding และ filtering ไฟล์มัลติมีเดียร์ video ที่อยู่ในรูปแบบของ command line ที่นำออกมาให้ใช้โดยอยู่ภายใต้ GNU General Public License ซึ่งเป็นเครื่องมือที่อยู่ในตระกูลเดียวกับ Mplayer สิ่งที่โครงงานฉบับนี้นำมาใช้ คือ การตัดไฟล์มัลติมีเดียให้เป็นไฟล์ย่อยๆขนาดเล็ก และ การแปลงรูปแบบของ codecs ให้เป็น MPEG-4 โดยคำสั่งที่ใช้มีดังนี้

- mencoder -ovc copy -oac copy -ss hh:mm:ss -endpos hh:mm:ss -o target from_file เป็นคำสั่งที่ใช้ในการตัดไฟล์วีดีโอใหญ่ๆให้กลายเป็นไฟล์ย่อยๆ โดย option -ss hh:mm:ss ใส่ค่าของเวลาเริ่มต้นในไฟล์ ส่วน –endpost hh:mm:ss ใส่ค่าของจำนวนเวลาที่นับจากเริ่มต้นที่ option –ss ว่าให้เป็นจำนวนเท่าใด

- mencoder file_input –vf harddup –ovc lavc –oac lavc –lavcopts vcodec=mpeg4:vqmax=4:acodec=ac3:abitrate=128 –of avi –o file_output เป็นคำสั่งที่ใช้ในการแปลงหรือ convert ไฟล์ให้เป็น MPEG-4 codecs โดยมี container เป็น avi

- cat firstfile.avi secondfile.avi > concatenatedfile.avi และคำสั่ง mencoder -o finalmovie.avi -noidx -oac copy -ovc copy concatenatedfile.avi ใช้ในการรวมไฟล์ที่ทำการเข้ารหัสเสร็จเรียบร้อยแล้วเข้าด้วยกันให้เป็นไฟล์เดียว

เมื่อทำการตัดไฟล์มัลติมีเดียร์ตัวอย่างให้เป็นไฟล์เล็กๆแล้วจึงทำการเขียน RSL ของงานแต่ละงานโดยภายใน RSL จะมีการเรียกให้ mencoder ทำงานแปลงไฟล์โดยใช้ GridFTP ส่งไฟล์ย่อยที่ทำการตัดไว้ไปรันบนเครื่องเป้าหมาย

บทที่ 3
องค์ประกอบของ Globus Toolkit 4.0.1

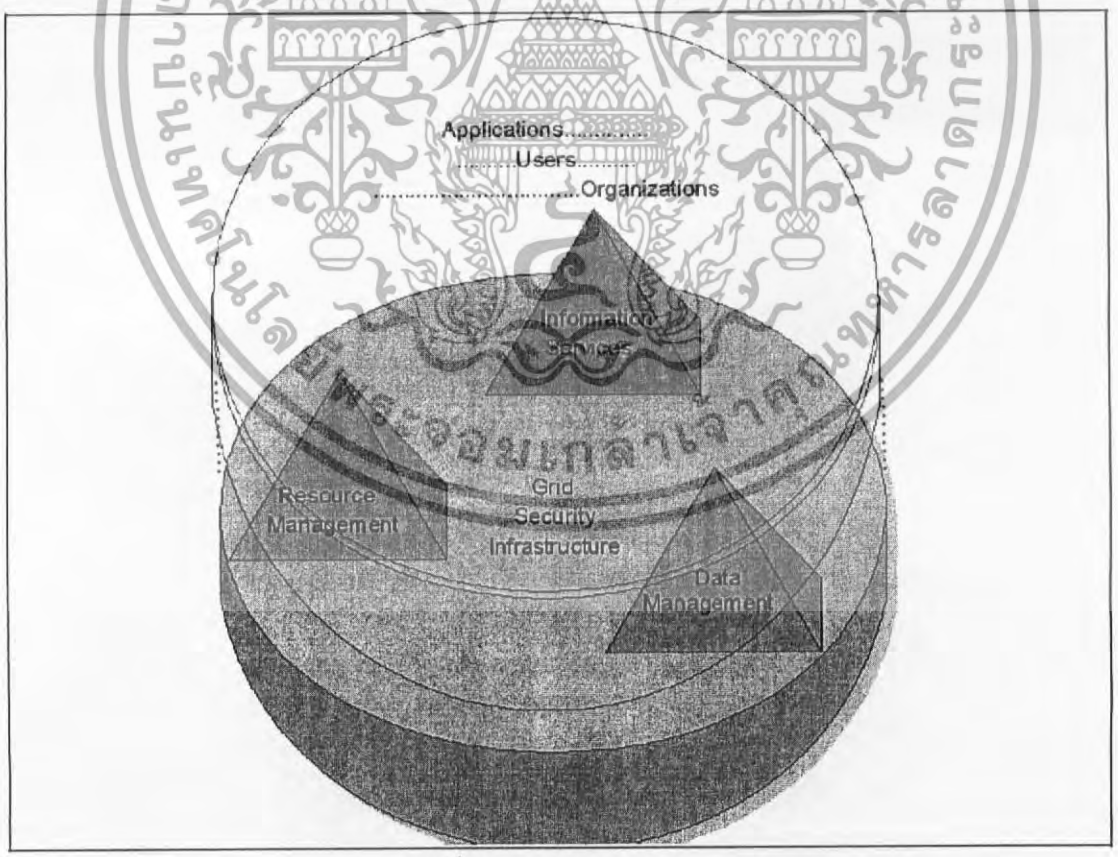
3.1. องค์ประกอบใน Three Pyramids

Globus Toolkit มี 3 พีระมิดของการสนับสนุนที่ถูกสร้างขึ้นชั้นบนสุดของโครงสร้างภายใน ของ Security คือ

1. Resource Management
2. Data Management
3. Information Services

ทั้ง 3 อย่างนี้ถูกสร้างอยู่ชั้นบนสุดของ GSI (Grid Security Infrastructure) ที่มีกระบวนการ ดังนี้

- single/mutual authentication การตรวจสอบสิทธิ์แบบเดี่ยวและแบบทั้ง 2 ฝ่าย
- confidential communication ความถูกต้องของข้อมูล
- authorization สิทธิ์ในการเข้าถึงข้อมูล
- delegation ตัวแทนในการใช้สิทธิ์



รูปที่ 3.1 Three Pyramids

3.1.1. Resource Management

ในส่วนนี้เป็นส่วนของการจัดการทรัพยากรต่างๆที่มีในระบบกริด ซึ่งมีทั้งการส่งงานไปประมวลผลที่เครื่องอื่นที่เราเรียกว่าการส่งงานแบบ remote มีการจัดทรัพยากรที่งานต้องการในการประมวลผล ซึ่งส่วนนี้จะสัมพันธ์กับส่วนจัดการข้อมูล (Data management) ซึ่งเป็นส่วนที่ทำหน้าที่ส่งไฟล์ไปยังเครื่องต่างๆ รวมไปถึงการอธิบายลักษณะของงานว่าเป็นงานชนิดใดรูปแบบใด

ส่วนของการจัดการทรัพยากรนี้จะมีเครื่องมือทาง scheduler จัดไว้ให้โดยความสามารถส่วนใหญ่จะอยู่ที่ภาษาที่อธิบายลักษณะงาน ซึ่งในตัว Globus toolkit เองนั้น ไม่มี scheduler ที่จัดการคลัสเตอร์จัดไว้ให้ มีเพียงเครื่องมือส่วนหนึ่งที่จัดการทางด้าน scheduler ให้นำไปใช้ได้

ส่วนจัดการทรัพยากรของสามพีระมิดนี้ เครื่องมือที่เป็นตัวแทนในการทำหน้าที่ที่สร้างขึ้นใน Globus toolkit คือ GRAM (Grid Resource Allocation Manager) ซึ่งจะกล่าวในส่วนถัดไป

3.1.2. Information Services

ส่วนของการจัดหาข้อมูลนี้ เป็นส่วนที่ทำหน้าที่ในการหาข้อมูลของทรัพยากรต่างๆในระบบว่าในระบบมีทรัพยากรเหลือให้งานนำไปใช้ได้เท่าใดและมีการตรวจสอบสถานะของระบบว่ามีงานที่รอการประมวลผลหรือกำลังทำการประมวลผลอยู่เท่าใด รวมไปถึงข้อมูลในเชิงคงที่และไม่คงที่ของระบบ

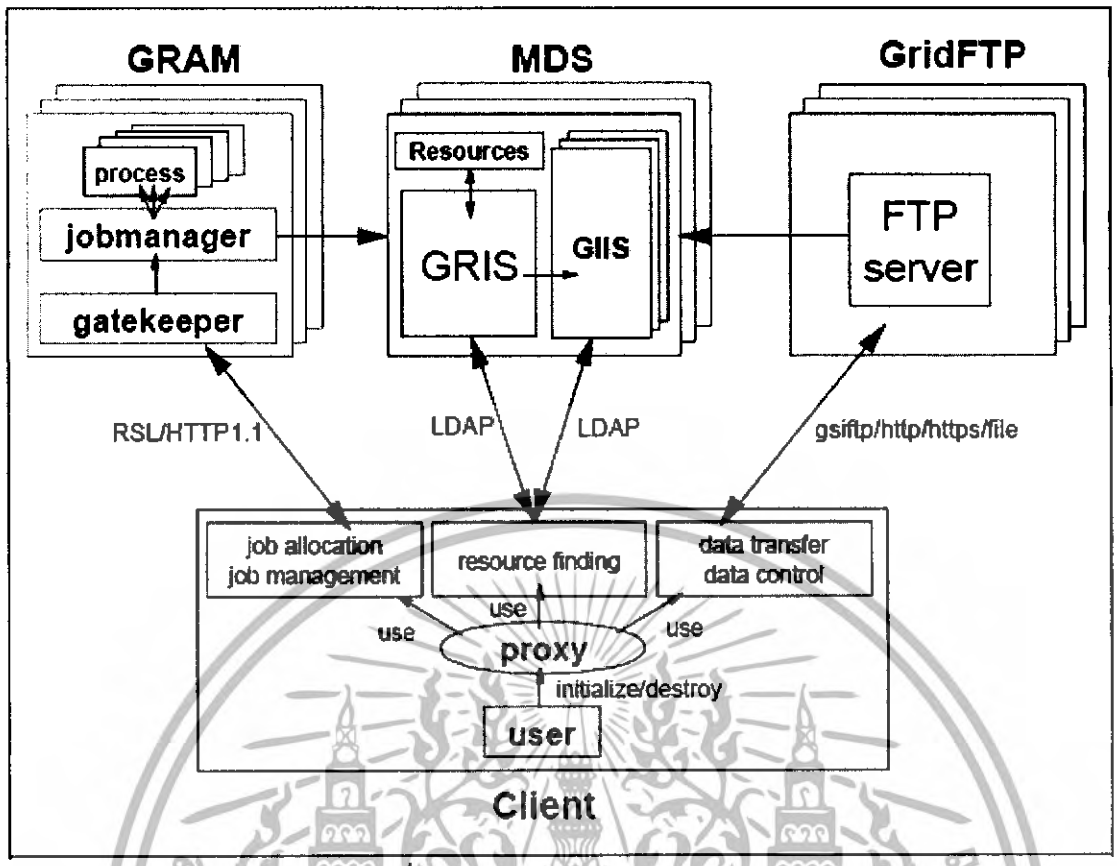
- ข้อมูลที่คงที่ เป็นข้อมูลแบบ static เช่น จำนวนของหน่วยความจำทั้งหมดของเครื่อง พื้นที่ว่างเหลือใน disk

- ข้อมูลไม่คงที่ เป็นข้อมูลแบบ dynamic เช่น เปอร์เซ็นต์การใช้งานของหน่วยประมวลผล หน่วยความจำที่เหลืออยู่ขณะนั้น

โดยภาพรวมของ Information Services นั้นคือการจัดหาข้อมูลของทรัพยากรทั้งหมดของระบบให้กับผู้ใช้เพื่อเป็นข้อมูลในการตัดสินใจในการส่งงานเข้าไปประมวลผล

3.1.3. Data management

ส่วนของการจัดการข้อมูลทั้งหมดซึ่งเน้นในเรื่องการส่งไฟล์ระหว่างเครื่องในระบบกริด ซึ่งจะใช้ GridFTP ที่สร้างขึ้นในเครื่องมือ Globus toolkit เป็นตัวแทนในการทำหน้าที่ทั้งหมด โดยส่วนนี้จะทำหน้าที่สัมพันธ์กับส่วนของการจัดการทรัพยากร ตัวอย่างเช่น เมื่องานที่ทำการส่งเข้าไปมีความต้องการในการใช้ไฟล์ในการประมวลผล ส่วนจัดการทรัพยากรจะแจ้งบอกให้เกิดการส่งไฟล์จากเครื่องที่เป็นแหล่งของไฟล์นั้นเข้ามาที่เครื่องประมวลผล หลังจากการประมวลผลเสร็จสิ้นผลลัพธ์จะถูกส่งกลับและไฟล์ที่ถูกส่งมาจะต้องทำการลบทิ้งไป

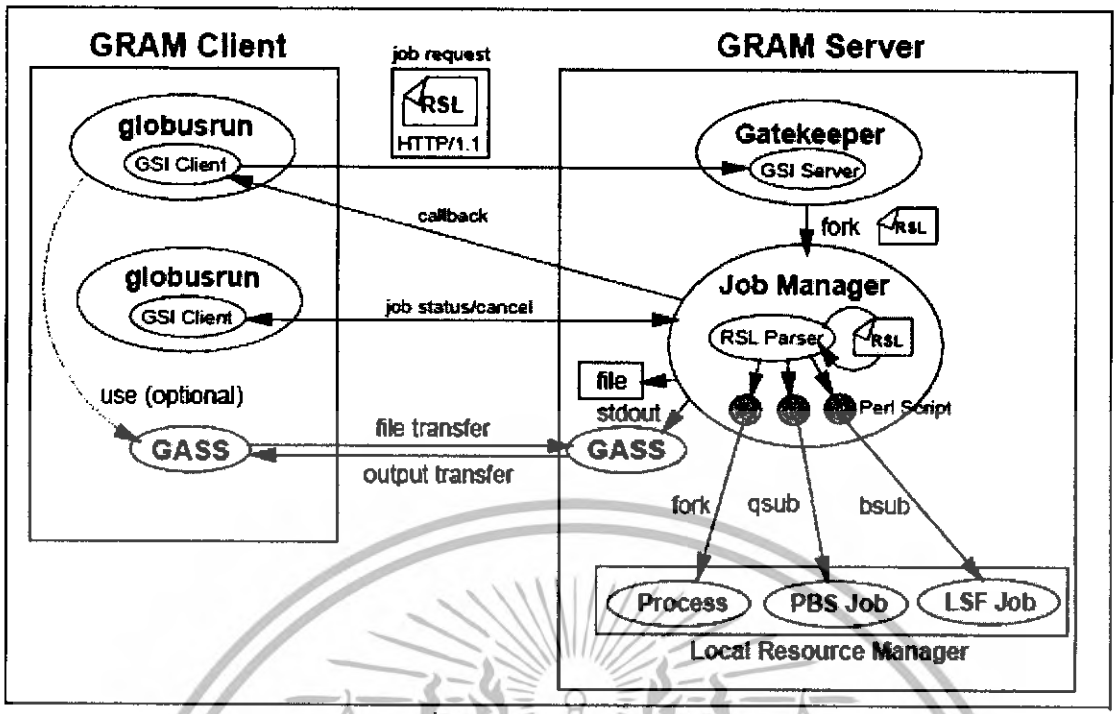


รูปที่ 3.2 ระบบโดยรวมของ Globus Toolkit

3.2. องค์ประกอบของ Globus toolkit ที่ใช้ประยุกต์ในพีระมิดทั้งสาม

3.2.1. GRAM (Grid Resource Allocation Manager)

เป็นองค์ประกอบหลักของ Resource Management Pyramids และเป็นโมดูลที่มีการประมวลผลแบบ remote มีการจัดการสถานะของการประมวลผล เมื่อ Client ทำการส่งงานเข้ามาคำร้องจะถูกส่งไปที่ remote host แล้วจะถูกจัดการโดย Gatekeeper daemon ที่อยู่ภายใน remote host นั้น จากนั้น Gatekeeper จะสร้างตัวจัดการงานนั้นเพื่อสั่งให้งานเริ่มประมวลผลและติดตามผลจากงานนั้น เมื่องานนั้นเสร็จสิ้นตัวจัดการงานจะส่งข้อมูลสถานะของงานกลับไปที่ Client แล้วจึงทำการสั่งให้งานที่เสร็จแล้วออกจากระบบ



รูปที่ 3.3 ภาพรวมของ GRAM

3.2.2. ส่วนประกอบหลักของ GRAM

3.2.2.1. คำสั่ง globusrun

globusrun เป็นคำสั่งใช้ในการส่งงานและจัดการงานแบบ remote คำสั่งนี้ถูกใช้ในเกือบทุกเครื่องมือที่เป็น GRAM clients และคำสั่งนี้มีการทำงานให้ดังต่อไปนี้

- การทำงานโดยการส่งคำร้องไปที่เครื่องที่เป็น remote โดยคำร้องนั้นเป็นคำร้องขอเพื่อส่งงาน

การส่งผ่านงานนั้นจะใช้ Security function(เช่น GSS-API) เพื่อตรวจสอบการใช้ mutual authentication ระหว่าง Server และ Client เพื่อตรวจสอบว่า Client นั้นมีสิทธิ์ที่จะส่งงานเข้าไปประมวลผล

- การทำงานโดยการส่งไฟล์ที่สามารถประมวลผลได้เข้าประมวลผลและส่งไฟล์ส่วนที่เก็บผลลัพธ์ของงานกลับไปที่ผู้ใช้ที่ทำการส่งงานมาประมวลผล

คำสั่ง globusrun ทำให้ได้ผลลัพธ์จากการประมวลผลงานจากเครื่องที่เป็น remote โดยที่มีการใช้ GASS เพื่อให้เกิด Security ในการทำให้การส่งไฟล์ระหว่างเครื่องใน grid นั้นปลอดภัย

นอกจากนี้ คำสั่ง globusrun ยังมีความสามารถในการเชื่อมต่อการส่งงานเข้าสู่ระบบคลัสเตอร์ที่สร้างไว้ด้วยโดยการใส่ตัวเลือกว่าระบบคลัสเตอร์นั้นเป็นรูปแบบใด เช่น PBS, Condor เป็นต้น

3.2.2.2. Resource Specification Language (RSL)

เป็นภาษาที่ถูกใช้โดย Client เพื่อส่งผ่านงาน โดยคำร้องส่งผ่านงานทั้งหมดจะอธิบายได้ด้วย RSL ซึ่งประกอบไปด้วย ไฟล์ที่ประมวลผลได้และสภาวะที่มันจะถูกประมวลผลเราสามารถชี้เฉพาะไปแต่ละเรื่องได้ เช่น ปริมาณ memory ที่ต้องการใช้ในการประมวลผลงานนั้นๆที่ เครื่องที่เป็น remote

Resource Specification Language เป็นภาษาที่ใช้อธิบายลักษณะงาน แล้วทำการวิเคราะห์หรือประมวลผลโดยใช้คำสั่ง Globusrun ที่เป็นเครื่องมือที่ Globus ในส่วนของ GRAM จัดไว้ให้ ในส่วนของ Globus 4.0.1 RSL จะเป็นภาษาที่พัฒนาขึ้นมาเป็น XRSL ซึ่งเป็นภาษาที่ใช้อธิบายลักษณะงานที่เปลี่ยนมาใช้ภาษา XML ในการเขียนอธิบาย ถ้าดูตามรุ่นของภาษา RSL ใน Globus version 3 ขึ้นไปนั้นจะเปลี่ยนรูปแบบของ format มาใช้ภาษา XML ในการเขียนทั้งหมด โดยตัวอย่าง Job Description เป็นดังต่อไปนี้

```
<?xml version="1.0" encoding="UTF-8"?>
<job>
   <executable>/bin/echo</executable>
   <directory>/tmp</directory>
   <argument>12</argument>
   <argument>abc</argument>
   <argument>34</argument>
   <argument>this is an example_string </argument>
   <argument>Globus was here</argument>
   <environment>
      <name>PI</name>
      <value>3.141</value>
   </environment>
   <stdin>/dev/null</stdin>
   <stdout>stdout</stdout>
   <stderr>stderr</stderr>
   <count>2</count>
</job>
```

ตัวอย่างนี้เป็นการสั่งให้ส่งงานที่ทำการแสดงผลออกหน้าจอด้วยคำสั่ง echo ไปที่เครื่อง remote แล้วส่งผลกลับมา เขียนไฟล์ stdout ที่อยู่ใน directory/tmp สิ่งที่เขียนลงไปจะอยู่ใน tag XML ที่ชื่อว่า <argument>

ผลลัพธ์ที่ออกมาที่ file stdout เป็น ดังนี้

12 abc 34 this is an example_string Globus was here

12 abc 34 this is an example_string Globus was here

ความสามารถส่วนใหญ่ของ GRAM จะอยู่ที่ภาษา RSL ในส่วนของ XML schema เป็นภาษาที่อธิบายภาษาสำหรับการขี้เฉพาะคำร้องของงาน โดยที่ ManagedJobService จะเป็นตัวแปลงภาษานี้ให้เป็นภาษาเฉพาะของ scheduler และ GRAM service จะเข้าใจกลุ่มของ elements ต่างๆ เช่น executable, arguments, directory

- RSL Elements สำหรับ GRAM

  <gram:executable> ใส่ path ของ program (type = rsl:pathType)

  <directory> Directory ที่จะใช้ run (type = rsl:pathType)

  <arguments> ใส่ string argument ของ program (type = rsl:argumentListType)

  <environment> ใส่ชื่อหรือค่าตัวแปร (type = gram:env_type)

  <stdin> (type = rsl:pathType)

  - เป็น stdin สำหรับโปรแกรม
  - A file path(แบบ absolute หรือ Relative) หรือ URL
  - ถ้า remote, file ทั้งหมดจะ pre-staged ก่อนการประมวลผล

  <stdout> (type = rsl:pathListType)

  - เป็น stdout ของ program
  - Multiple file paths หรือ URL
  - ถ้าใช้แบบ remote, file จะมีการส่งแบบเพิ่มจำนวนขึ้นเรื่อยๆ

  <stderr> (type = rsl:pathListType)

  - เป็น stderr ของโปรแกรม
  - Multiple File Path หรือ URL
  - ถ้าใช้แบบ remote, file จะมีการส่งแบบเพิ่มจำนวนขึ้นเรื่อยๆ

  <count> (type = rsl:integerType)

  - ใส่จำนวน process ที่ต้องการใช้รัน

  <hostCount> (type = rsl:integerType)

- บน SMP multi-computers จะใส่จำนวน node ที่จะใช้เพื่อกระจายจำนวน process ที่ใส่ใน count element

<project> (type = rsl:integerType)

- *Project (account) against which to charge*

<queue> (type = rsl:stringType)

- ชื่อของ queue ใน scheduler ที่จะทำการส่งงานเข้าไป

<maxWallTime> (type = rsl:*long*Type)

- ขนาด wall clock runtime สูงสุดในหน่วยนาที เป็นเวลาที่งานจะใช้ในการประมวลผล ถ้าเกินกว่านี้งานจะถูกนำออกจากการประมวลผล

<maxCpuTime> (type = rsl:longType)

- ขนาด CPU runtime สูงสุดในหน่วยนาที

<maxTime> (type = rsl:longType)

- ใช้ประยุกต์ได้ก็ต่อเมื่อ maxWallTime และ maxCpuTime ไม่ได้ถูกใช้
- ขนาด Wall Clock หรือ CPU Runtime สูงสุดในหน่วยนาทีซึ่งเป็นตัวเลือกของ scheduler

<maxMemory> (type = rsl:integerType)

- ขนาด memory สูงสุดของแต่ละ process ในหน่วย megabytes

<minMemory> (type = rsl:integerType)

- ขนาด memory ต่ำสุดของแต่ละ process ในหน่วย megabytes

<jobType> (type = rsl:jobRunType)

- ใส่ชนิดของ job ที่จะรัน ซึ่งมีดังนี้ "mpi","single","multiple","condor"
  * "mpi" รันโปรแกรมโดยใช้คำสั่ง "mpirun –np <count>"
  * "single" รัน single instance ของ program แล้วทำให้โปรแกรม start process หรือ threads ที่เหลือ ซึ่งเป็นจำนวน count-1 ชนิดของ job ชนิดนี้ ดีสำหรับ scripts และ สำหรับ multithreads programs
  * "multiple" ใช้ start instance ของโปรแกรมในจำนวนที่ใส่ใน element count โดยใช้ กระบวนการของ scheduler ที่เหมาะสม
  * "condor" start condor process จำนวน count รันใน standard universe (link ไปที่ condor library สำหรับ remote I/O, checkpoint/restart, etc)

<scratchDir> (type = rsl:pathType)

- ใส่ sub directory ที่อยู่ใน <path> ที่ถูกสร้างขึ้นให้ job
- path นี้ถ้ามีการเกี่ยวข้อง จะเกี่ยวกับ 2 สิ่ง
  1) a site ที่มีการconfig scratch directory
  2) Users HOME directory บน JM host

<gassCache> (type = rsl:pathType)

- Override the default GASS cache directory
- default นั้นคือส่วนของ site ที่สามารถ config ได้ หรือเป็น path ~/.globus/.gasscache ถ้าไม่ได้ config

<libraryPath> (type = rsl:pathListType)

- ตั้งค่าของ job environment ดังนั้น application ที่สร้างโดยใช้ shared library จะสามารถรันได้อย่างคล่องตัว

<fileStageIn> (type = gram:fileStageInType)

- ใส่รายการของ remote url ให้ local file เพื่อให้ไปที่ host ที่ซึ่ง job จะทำการรัน

<fileStagedInShared> (type = gram:fileStageInType)

- รายการไฟล์ที่จะไปตั้งที่ GASS cache
- Links จาก cache ไปที่ local file ที่จะทำการสร้างขึ้น

<fileStagedOut> (type = gram:fileStageOutType)

- รายการไฟล์ที่จะออกมาเมื่อ job process เสร็จสมบูรณ์

3.2.2.3. The gatekeeper daemon

Gatekeeper daemon จะสร้างความปลอดภัยในการสื่อสารระหว่าง Client และ Server ซึ่งคล้ายๆกับ inetd daemon ในเชิงของฟังก์ชันการทำงาน Gatekeeper จะติดต่อกับ Client ของ GRAMและตรวจสอบสิทธิ์ของผู้ที่จะส่งผ่านงานที่ถูกต้อง หลังการตรวจสอบสิทธิ์ gatekeeper จะสร้าง Job manager มาเป็นตัวแทนในการตรวจสอบระดับสิทธิ์เพื่อที่จะติดต่อกับ Clients

3.2.2.4. The job manager

Job manager ถูกสร้างโดย gatekeeper daemon เป็นส่วนหนึ่งของกระบวนการในการส่ง job request ซึ่ง job manager จะให้การ interface ที่ควบคุมการจัดสรรปันส่วนของ resource manager ของแต่ละแห่ง เช่น ควบคุม scheduler PBS,LSF หรือ LoadLeveler

การทำงานต่างๆใน job manager มีดังนี้

- ระบบวิเคราะห์ภาษาที่อธิบายลักษณะงานและ RSL Script
- ระบบการจัดส่วนของคำร้องของงานไปให้ตัวจัดการทรัพยากรแต่ละแห่ง

- การทำการส่ง callback กลับไปให้ Client ถ้าจำเป็น

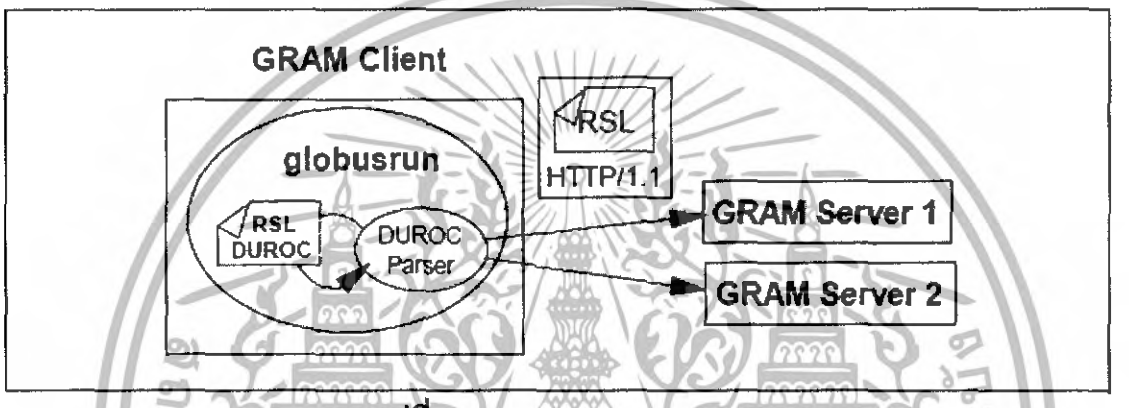
- การทำการรับสถานะและยกเลิกคำร้องจาก Client

- การทำงานส่งผลลัพธ์เป็น Output กลับไปให้ Client โดยใช้ GASS

3.2.2.5. Global Access to Secondary Storage (GASS)

GRAM ใช้กระบวนการส่ง output file จาก server ไป Client ของ GASS โดยมีฟังก์ชันบางอันใน API ที่อยู่ภายใต้ GSI protocol นำมาใช้เพื่อความปลอดภัยในการส่งไฟล์ ซึ่งในกระบวนการดังกล่าวนี้ถูกใช้โดยคำสั่ง globusrun, gatekeeper, และ job manager

3.2.2.6. Dynamically-Updated Request Online Coallocator (DUROC)



รูปที่ 3.4 ภาพรวมของ DUROC

ถ้าใช้กระบวนการ DUROC ผู้ใช้สามารถส่งผ่านงานไปยัง job manager ตัวอื่นที่ host อื่นๆ หรือ ส่งผ่านงานไปยัง job manager ตัวอื่นที่อยู่ใน host เดียวกันก็ได้

RSL script ที่บรรจุ syntax ของ DUROC จะถูกวิเคราะห์ที่ GRAM Client และไปหา job manager ในที่ต่างๆกัน

3.2.3. Monitoring & Discovering Service (MDS)

Monitoring and Discovering Service (MDS) โดยหลักแล้วจะมีความเกี่ยวพันกับการจัดเรียง, การกระจาย, การเก็บข้อมูล และการประมวลผลของข้อมูลต่างๆ เกี่ยวกับสถานะของ resource, service, system configuration ข้อมูลต่างๆที่เก็บมานั้นจะถูกนำไปใช้สำหรับการค้นหา resource หรือ service ใหม่ๆ หรือ เพื่อใช้ในการเฝ้าสังเกตสถานะของระบบ

GT4 ได้จัดเตรียม WS-RF และ WS-Notification ซึ่งเป็นไปตามระบบ MDS ซึ่งในที่นี้จะเรียกว่า MDS4

คุณสมบัติของทรัพยากรที่ถูกจัดเตรียมโดย WS-RF สามารถลงทะเบียนกับ MDS4 สำหรับการเก็บข้อมูล WS-RF service ได้จัดการคุณสมบัติต่างๆเช่น GRAM และ RFT ซึ่งในขณะที่เรียกใช้ GT4 container นั้น service นี้ก็จะถูกลงทะเบียนไปยัง MDS4 ด้วย

MDS4 นั้นประกอบไปด้วย 2 service ใหญ่ๆ ก็คือ Index Service และ Trigger Service ซึ่งมีพื้นฐานบน Aggregator Framework ซึ่งจะอธิบายในเรื่องถัดไป

3.2.3.1. Index Service

Index Service นั้นเป็นส่วนประกอบหลักของ GT4 MDS ซึ่ง GT4 container ทั้งหมดจะมี default indexing service (DefaultIndexService) เช่น WSRF service โดยที่ Index service นั้นกระทำกับข้อมูลต่างๆผ่านคุณสมบัติมาตรฐานของ WS-RF resource และ subscription/notification (WS-ResourceProperty และ WS-BaseNotification) Index Service นั้นมีความเป็นไปได้ที่จะสามารถเก็บข้อมูลจากหลายๆที่มารวมกันไว้เพียงที่เดียว ซึ่งการลงทะเบียน WSRF ต่างๆพร้อมทั้ง Index service นั้นจะถูกเก็บรักษาในรูปของ Service Group Entries โดย Index service ซึ่งข้อมูลที่เก็บอยู่ใน Index service นั้นสามารถ query โดยการใช้ XPath query

จุดเด่นของ Index service

- Index service สามารถปรับให้มีการจัดลำดับได้ แต่ ไม่มี global index ใดๆที่จัดการข้อมูลเกี่ยวกับทุกๆ resource บนระบบ Grid ได้
- การนำเสนอ resource ใน Index service ไม่รับประกันเกี่ยวกับ resource ที่มีอยู่สำหรับผู้ใช้ Index นั้นๆ
- การลงทะเบียนแต่ละครั้งไปยัง Index service นั้นมีวันหมดอายุและต้องการการลงทะเบียนใหม่เป็นช่วงๆเพื่อทำให้รู้ว่ายังคงมี resource หรือ service นั้นๆอยู่

3.2.3.2. Trigger Service

Trigger Service จะเก็บข้อมูลและเปรียบเทียบข้อมูลเหล่านั้นกับกลุ่มของเงื่อนไขที่ถูกระบุไว้ใน configuration file ซึ่งเงื่อนไขจะต้องระบุเป็น XPath expression

ไฟล์ที่ 3.1. Trigger Service registration & configuration file

```
$GLOBUS_LOCATION/etc/globus_wsrf_mds_trigger/trigger_registration_aggregation.xml
```

Configuration file นี้กำหนดกฎเกณฑ์ในรูปของ XPath query expression ตัวอย่างดังต่อไปนี้

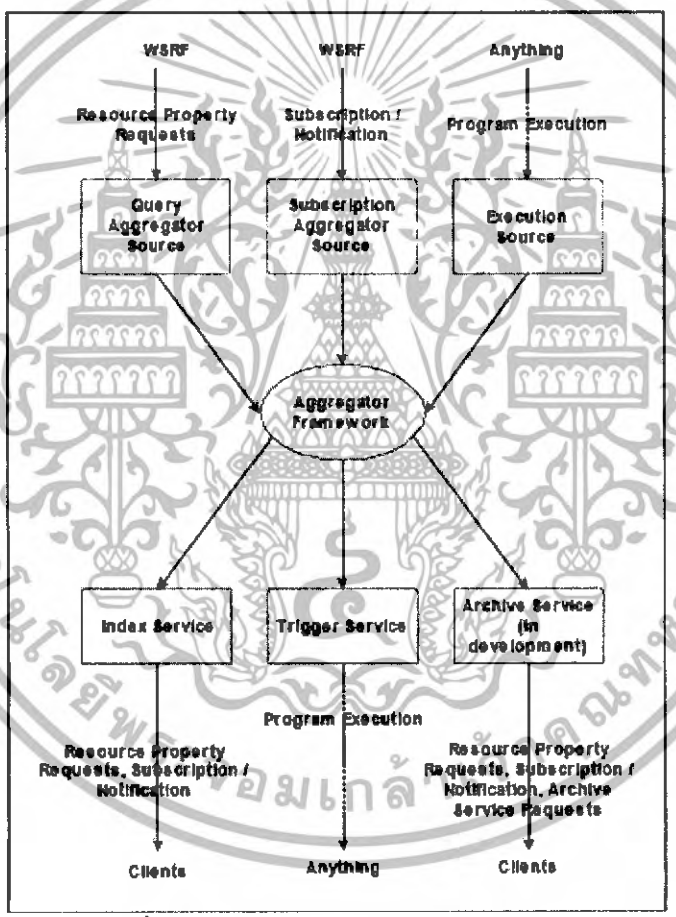
ตัวอย่างที่ 3.1. ตัวอย่างการลงทะเบียน Trigger Service

```
<trigger:TriggerRuleType>
        <trigger:matchingRule>//*[local-name()='GLUECE']</trigger:matchingRule>
        <trigger:actionScript>glue-trigger-action.sh</trigger:actionScript>
        <trigger:minimumFiringInterval>600</trigger:minimumFiringInterval>
</trigger:TriggerRuleType>
```

3.2.3.3. Aggregator Framework

MDS Index service และ MDS Trigger Service นั้นพิเศษสำหรับ Aggregator Framework ซึ่ง Aggregator Framework เป็น software framework ที่ใช้สำหรับสร้าง software service ซึ่งเก็บและรวบรวมข้อมูล ซึ่งเรียกว่า "aggregator services"

Aggregator Service นั้นทำหน้าที่ในการเก็บข้อมูลจากหนึ่งในสามชนิดของ aggregator source เช่น query source ที่ใช้กระบวนการ WS-Resourceproperty ในการเก็บข้อมูล, subscription source ที่ใช้กระบวนการ WS-Notification subscription/notification ในการเก็บข้อมูล, หรือ execution source ซึ่งประมวลผล administrator-provide ในการเก็บข้อมูลในรูปแบบของ XML



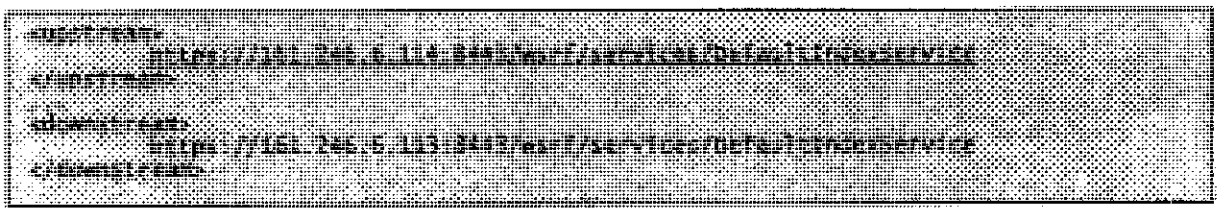
รูปที่ 3.5 MDS4 Aggregator Framework

ไฟล์ที่ 3.2. MDS Hierarchy file

```
$GLOBUS_LOCATION/etc/globus_wsrf_mds_index/hierarchy.xml
```

Hierarchy file นี้จะระบุ URLs สำหรับ Upstream และ Downstream ของ Index Service ตัวอย่างการกำหนด hostname, port และระบุ Default Index Service ดังตัวอย่างที่...

ตัวอย่างที่ 3.2. ตัวอย่างการกำหนด Upstream & Downstream service



3.2.3.4. MDS resource attribute

จากที่ได้กล่าวมาผู้ใช้สามารถ query เพื่อดู resource attribute ซึ่งจะแสดงให้เห็นถึงการปรับแก้ และ สถานะของ resource ข้อมูลใน Globus MDS นั้นถูกเก็บโดยรวมค่า timeout ไว้กับข้อมูล ซึ่งค่านี้เปรียบได้กับ lifetime ทุกๆข้อมูลที่อยู่ใน MDS Index Service จะมี lifetime และข้อมูลจะหมดอายุหากไม่มีการ update ในช่วงเวลาขณะหนึ่งซึ่งถูกระบุโดยค่า lifetime

3.2.3.5. MDS resource property

ข้อมูลจาก Globus MDS Index Service ถูกดูแลเหมือนกับ Web Service Resource Framework (WSRF)

ตัว client สามารถใช้ WSRF get -property และ WS-Notification subscribe เพื่อใช้ในการ query ข้อมูล หรือเพื่อประกาศการเปลี่ยนแปลงข้อมูลต่างๆ ซึ่งข้อมูลนั้นสามารถ query ด้วยคำสั่ง wsrf-get-property หรือ wsrf-qurey ด้วยคำสั่งดังนี้

- wsrf-query [options] [query expression] [dialect]

ตารางที่ 3.1 option ของ wsrf-query

-h, --help	แสดงข้อมูลช่วยเกี่ยวกับคำสั่ง
-d, --debug	ใช้ debug mode.
-e, --eprFile <file>	ระบุ *XML* file ที่เก็บจุดอ้างอิง *WS-Addressing*
-s, --service <url>	ระบุ service URL
-k, --key <name value>	ระบุ resource key. โดยที่ name คือ QName ของ resource key ในรูปแบบของ string: {namespaceURI}localPart, ซึ่ง value คือค่าตัวอย่างของ key
-f, --descriptor <file>	ระบุ file ที่กล่าวถึง client security.
-a, --anonymous	ใช้ anonymous authentication. สนับสนุนเฉพาะการส่งแบบ security หรือกระบวนการ GSI Secure Conversation authentication.

-g, --delegation <mode>	ใช้ delegation. mode สามารถเป็นได้ระหว่าง 'limited' หรือ 'full'. สนับสนุนเฉพาะกระบวนการ GSI Secure Conversation authentication
-l, --contextLifetime <value>	กำหนด lifetime ของ client security context. ค่า value มีหน่วยเป็น milliseconds. สนับสนุนเฉพาะกระบวนการ GSI Secure Conversation authentication
-m, --securityMech <type>	ระบุกระบวนการทำ authentication . type สามารถเป็น 'msg' สำหรับ GSI Secure Message, หรือ 'conv' สำหรับ GSI Secure Conversation
-c, --serverCertificate <file>	ระบุ server's certificate file ที่ใช้สำหรับการ encryption. ต้องระบุหากใช้กระบวนการ GSI Secure Message authentication
-p, --protection <type>	ระบุระดับของการป้องกัน type สามารถเป็น 'sig' สำหรับ signature หรือ 'enc' สำหรับ encryption.
-z, --authorization <type>	ระบุชนิดของการ authorization. type สามารถเป็น 'self', 'host', 'none', หรือ ข้อความใดๆที่ระบุไปยัง remote party.

ตัวอย่าง :

- wsrf-get-property [options] <property>

โดยที่ <property> เป็น Qname ของ resource property ในรูปของ :

{namespaceURI}localPart.

ตารางที่ 3.2. option ของ wsrf-get-property

-h, --help	แสดงข้อมูลช่วยเกี่ยวกับคำสั่ง
-d, --debug	ใช้ debug mode.
-e, --eprFile <file>	ระบุ *XML* file ที่เก็บจุดอ้างอิง *WS-Addressing*
-s, --service <url>	ระบุ service URL
-k, --key <name value>	ระบุ resource key. โดยที่ name คือ QName ของ resource key ในรูปแบบของ string: {namespaceURI}localPart, ซึ่ง value คือค่าตัวอย่างของ key
-f, --descriptor <file>	ระบุ file ที่กล่าวถึง client security.
-a, --anonymous	ใช้ anonymous authentication. สนับสนุนเฉพาะการส่งแบบ security หรือกระบวนการ GSI Secure Conversation authentication.
-g, --delegation <mode>	ใช้ delegation. mode สามารถเป็นได้ระหว่าง 'limited' หรือ 'full'. สนับสนุนเฉพาะกระบวนการ GSI Secure Conversation authentication
-l, --contextLifetime <value>	กำหนด lifetime ของ client security context. ค่า value มีหน่วยเป็น milliseconds. สนับสนุนเฉพาะกระบวนการ GSI Secure Conversation authentication
-m, --securityMech <type>	ระบุกระบวนการทำ authentication . type สามารถเป็น 'msg' สำหรับ GSI Secure Message, หรือ 'conv' สำหรับ GSI Secure Conversation
-c, --serverCertificate <file>	ระบุ server's certificate file ที่ใช้สำหรับการ encryption. ต้องระบุหากใช้กระบวนการ GSI Secure Message authentication
-p, --protection <type>	ระบุระดับของการป้องกัน type สามารถเป็น 'sig' สำหรับ signature หรือ 'enc' สำหรับ encryption.
-z, --authorization <type>	ระบุชนิดของการ authorization. type สามารถเป็น 'self', 'host', 'none', หรือ ข้อความใดๆที่ระบุไปยัง remote party.

ตัวอย่าง :

```
[nodeC@Test3]$ wsrf-get-property -s http://localhost:8080/wsrf/services/CounterService \
        -k "{http://counter.com}CounterKey" 123 \
        "{http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceLifetime-1.2-draft-01.xsd}CurrentTime"
```

3.2.3.6. Monitoring Service

จากที่ได้กล่าวมาเราได้พูดในส่วนของ physical resource แต่ MDS สามารถเฝ้าสังเกตการณ์ปรับแก้ และ สถานะข้อมูลของ service. MDS Index Service เก็บข้อมูลเกี่ยวกับ service อื่นๆที่ทำงานอยู่บน container ซึ่งข้อมูลของ service ที่ระบุจะถูกเก็บเมื่อลงทะเบียนกับ Aggregator Service โดยที่สามารถลงทะเบียน service โดยใช้คำสั่ง mds-servicegroup-add ซึ่งการลงทะเบียนแต่ละครั้งนั้นมี lifetime และถ้าหากว่า service ไม่ได้ทำการลงทะเบียนซ้ำในช่วงเวลาที่กำหนดก็จะทำให้ service นั้นหมดอายุ

MDS Aggregator Service เก็บข้อมูลจากทุกๆ service ที่ลงทะเบียนโดยที่ Globus Toolkit จัดเตรียมตัวอย่างเพื่อแสดงให้เห็นว่าลงทะเบียน service กับ MDS Aggregator Service อย่างไร

ไฟล์ที่ 3.3. MDS Aggregation registration file

```
$GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-
registration.xml
```

ตัวอย่าง Aggregator file นี้กำหนดเกี่ยวกับการลงทะเบียน service ซึ่งการลงทะเบียนแต่ละครั้งจะระบุ grid resource, service group และ service configuration parameter ซึ่ง file นี้จะถูกใช้โดยคำสั่ง mds-servicegroup-add เพื่อดูแลการลงทะเบียนระหว่าง grid resource และ Index Service

ตัวอย่างที่ 3.3 Index Service URL ถูกกำหนดเป็น default Service Group End Point Reference

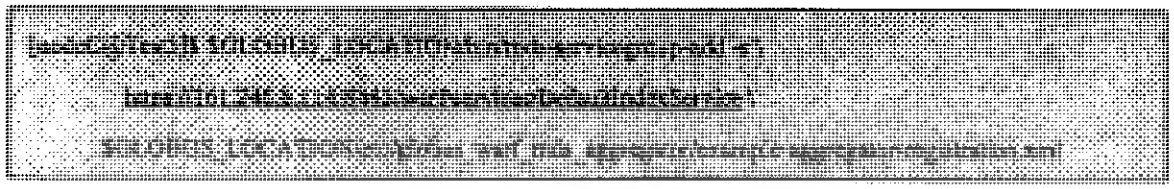
- mds-servicegroup-add [options] *config.xml*

ตารางที่ 3.3 option ของ mds-servicegroup-add

-s <URL>	สำหรับ option นี้เป็นส่วนสำคัญที่จำเป็นต้องมีทุกครั้งที่มีการใช้คำสั่ง mds-servicegroup-add ที่ระบุไปยัง url ปลายทางที่มี DefaultIndexService อยู่
-a	สำหรับการเชื่อมต่อแบบ anonymous ในกรณีที่ไม่มีการทำ certificate
-z, --authorization <type>	ระบุชนิดของการ authorization. type สามารถเป็น 'self', 'host', 'none'

config.xml	Path ที่ระบุไปยัง configuration file. ตัวอย่างที่มีให้ใน Globus Toolkit $GLOBUS_LOCATION/etc/globus_wsrf_mds_aggregator/example-aggregator-registration.xml

ตัวอย่างที่ 3.4. ตัวอย่างการใช้คำสั่ง mds-servicegroup-add



ตารางที่ 3.4. การเปลี่ยนแปลงจาก MDS2 ➔ MDS4

Description	GT2 Version	GT4 Version
ประเภทของตัวอธิบายข้อมูลทรัพยากร	การตัดลำดับข้อมูลด้วย LDAP	เอกสารข้อมูลประเภท XML
ภาษาที่ใช้ในการ Query	LDAP queries	XPath queries
protocol ในการติดต่อสำหรับการ queries	LDAP	WS-ResourceProperties
APIs ที่ใช้สำหรับการ queries	LDAP APIs	WS Core APIs
คำสั่งที่ใช้สำหรับ queries จาก client	grid-info-search	wsrf-get-property, wsrf-get-properties, wsrf-query
GUI ที่เปิดให้ใช้งาน	LDAP browsers	WebMDS
protocol ในการติดต่อสำหรับการ	ไม่สนับสนุน	WS-Notification

Description	GT2 Version	GT4 Version
subscription/notification		
APIs ที่ใช้ในการ subscription/notification	ไม่สนับสนุน	WS Core APIs
การสนับสนุนความปลอดภัย	ความปลอดภัยมาตรฐาน SAML ใช้ X.509 user, proxy และ host certificates	ความปลอดภัยมาตรฐาน HTTPS ใช้ X.509 user, proxy และ host certificates
Queryable index of aggregated information	GIIS, ซึ่งประกาศข้อมูลโดยใช้มาตรฐาน LDAP ดังที่ได้กล่าวมา	WS MDS Index Server, ซึ่งประกาศข้อมูลโดยใช้มาตรฐาน WSRF ดังที่ได้กล่าวมา
Queryable source of non-aggregated information	GRIS, ซึ่งใช้ information providers ในการรวบรวมข้อมูลจาก service และประกาศข้อมูลเหล่านั้นผ่านทาง LDAP ดังที่ได้กล่าวมา	web services, ซึ่งประกาศข้อมูลเกี่ยวกับทรัพยากรของตัวเองโดยใช้มาตรฐาน WSRF ดังที่ได้กล่าวมา
กระบวนการ Index registration	MDS servers (GRIS's และในบางกรณีของ GIIS's) ลงทะเบียนตนเองด้วย GIIS MDS server ถูกปรับให้ลงทะเบียนตัวเองไปยัง remote index โดยแก้ไขไฟล์ grid-info-resource-register.conf ใน MDS server's ที่จัดเตรียมข้อมูลเกี่ยวกับตำแหน่งของ remote index ในการ	WS MDS Index servers ดูแลกลุ่มของ service ที่ลงทะเบียนข้อมูล (ค่า timeout , กระบวนการที่ใช้ในการนำข้อมูล, และกระบวนการเฉพาะ เพิ่มเติม) การลงทะเบียนจะสำเร็จได้โดยเพิ่มสิทธิ์ให้กับกลุ่มของ service ผ่านคำสั่ง mds-servicegroup-add

Description	GT2 Version	GT4 Version
	ลงทะเบียน และค่า timeout สำหรับการลงทะเบียน	
กระบวนการที่ index ใช้ในการเก็บข้อมูลต่างๆ	GIIS's ส่ง LDAP queries ไปยัง remote server.	WS MDS Index servers ใช้พื้นฐานโครงสร้าง plugin เพื่อสนับสนุนบางกระบวนการในการเก็บข้อมูล Globus Toolkit จัดหา plugin ซึ่งสนับสนุนการเก็บข้อมูลผ่านการลงคะแนน (การ queries คุณสมบัติทรัพยากร), subscription/notification, และโดยการประมวลผล.

ตารางที่ 3.5. การเปลี่ยนแปลงจาก MDS3 ➔ MDS4

Description	GT3 Version	GT4 Version
การทำงาน Query	FindServiceData (เพื่อเก็บข้อมูลเดียวจาก service โดยชื่อ หรือ เพื่อดำเนินการทำ Xpath query เทียบกับ service ของข้อมูล)	GetResourceProperty (เพื่อเก็บข้อมูลคุณสมบัติทรัพยากรเพียงตัวเดียวโดยชื่อ), GetMultipleResourceProperties (เพื่อเก็บข้อมูลคุณสมบัติทรัพยากรหลายตัวโดยชื่อ), และ QueryResourceProperties (เพื่อดำเนินการ Xpath query เทียบกับ service ของทรัพยากร).
APIs ที่ใช้สำหรับการ queries	OGSI (GT3) Core APIs	WS Core APIs

Description	GT3 Version	GT4 Version
คำสั่งที่ใช้สำหรับ queries จาก client	ogsi-find-service-data	wsrf-get-property, wsrf-get-properties, wsrf-query
GUI ที่เปิดให้ใช้งาน	globus-sdb (standalone client) and WebSDB (web interface)	WebMDS (web interface)
การทำงานสำหรับ subscription/notification	OGSI NotificationSource / NotificationSink	WS-Notification
APIs ที่ใช้สำหรับ subscription/notification	OGSI (GT3) Core APIs	WS Core APIs
กระบวนการ Index registration	GT3 services สามารถปรับแต่งให้ประกาศข้อมูล service ไปยัง index services.	WS MDS Index servers ดูแลกลุ่มของ service ที่ลงทะเบียนข้อมูล (ค่า timeout , กระบวนการที่ใช้ในการนำข้อมูล, และกระบวนการเฉพาะ เพิ่มเติม) การลงทะเบียนจะสำเร็จได้โดยเพิ่มสิทธิ์ให้กับกลุ่มของ service ผ่านคำสั่ง mds-servicegroup-add

3.2.3.7. ทดสอบ MDS4

หลังจากที่ทำการลง Globus Toolkit 4.0.1 พร้อมทั้ง MDS เสร็จเรียบร้อยแล้ว ทำการทดสอบการทำงานของ MDS4 ดังนี้

1. login เข้าสู่เครื่องด้วย Globus user
2. สั่ง start container
3. ในอีกหน้าต่างหนึ่งให้ทำการ login เข้าใช้ user ที่มีการทำ certificate ของระบบ

4. พิมพ์ command ดัง *ตัวอย่างที่ 5* แล้วสังเกตผลที่ได้

ตัวอย่างที่ 3.5.

```
[nodeC@Test3]$ wsrf-query -s \
http://161.246.6.114:8443/wsrf/services/DefaultIndexService "/*"
```

3.2.4. GridFTP

GridFTP ทำให้การส่งข้อมูลระหว่างเครื่องต่างๆในระบบ grid มีความปลอดภัยมากขึ้น คำว่า GridFTP จะหมายถึง protocol, server หรือ ส่วนของเครื่องมือก็ได้

3.2.4.1. GridFTP protocol

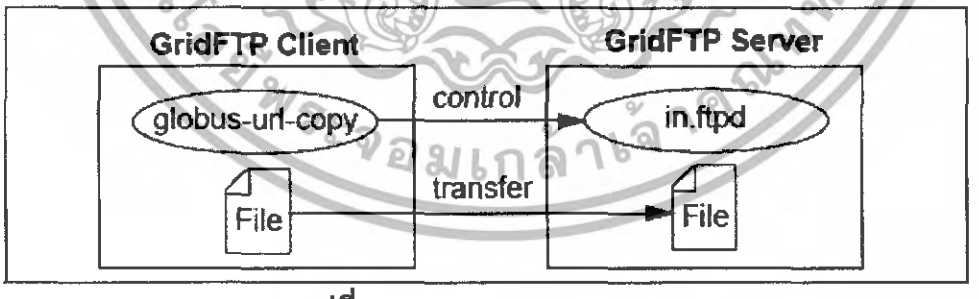
GridFTP เป็น protocol ที่จะใช้ในการส่งข้อมูลทั้งหมดบน grid ซึ่งมีพื้นฐานมาจาก FTP แต่เพิ่มขยายมากกว่า protocol มาตรฐานปกติ เช่น multistreamed transfer, auto tuning และความปลอดภัยพื้นฐานของ globus ซึ่ง protocol นี้ยังอยู่ในระหว่างการร่างโครงสร้าง

3.2.4.2. GridFTP server and client

Globus toolkit มี GridFTP Server และ GridFTP Client ที่สร้างโดย in.ftpd และโดยใช้คำสั่ง globus-url-copy

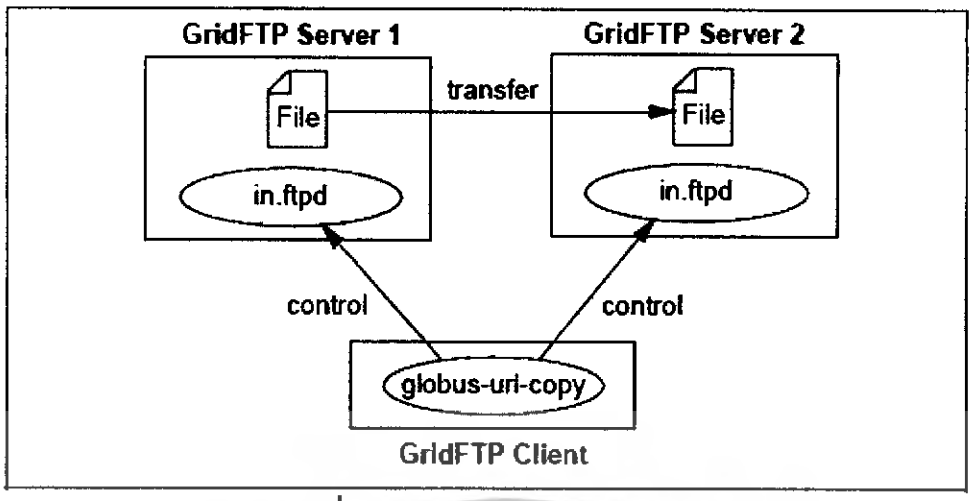
GridFTP server และ client สนับสนุน file transfer 2 ประเภท คือ แบบ standard และแบบ third-party

- Standard file transfer เป็นแบบที่ client ส่ง local file ไปที่เครื่องที่เป็น remote ที่เป็นเครื่องที่รัน FTP server



รูปที่ 3.6 standard file transfer

- Third-party file transfer เป็นแบบที่มีไฟล์ใหญ่ๆ 1 ไฟล์อยู่ใน storage ของเครื่องที่เป็น remote และ Client ต้องการที่จะ copy ไฟล์นั้นไปที่ remote server เครื่องอื่น



รูปที่ 3.7 Third-party file transfer

3.2.4.3. GridFTP tools

Globus toolkit มีส่วนของเครื่องมือที่สนับสนุนการส่งข้อมูลด้วย GridFTP โดยมี gsi-ncftp package ที่เป็นตัวหนึ่งของเครื่องมือที่ใช้ติดต่อด้วย GridFTP Server

GASS API package ก็เป็นส่วนหนึ่งของเครื่องมือทาง GridFTP ซึ่งจะถูกใช้โดย GRAM เพื่อส่งไฟล์ผลลัพธ์จาก server ไป client

GridFTP เป็นกุญแจสำคัญเพื่อความปลอดภัยและการแสดงผลของการส่งข้อมูล

ทั้ง 3 อย่างนี้รวมอยู่ในสามพีระมิดซึ่งเป็นรูปแบบของโมดูลที่สามารถทำงานแยกเป็นอิสระจากกัน แต่อย่างไรก็ตามแต่ละส่วนก็ต่างเป็นองค์ประกอบที่ทำให้ทุกส่วนสมบูรณ์

3.2.4.4. GSI – Grid Security Infrastructure

มีระบบความปลอดภัยดังที่กล่าวไว้ข้างต้น ในส่วนของโครงสร้างภายใน มีพื้นฐานมาจาก SSL Protocol (Secure Socket Layer), Public Key encryption และ x.509 Certificates

สำหรับในการทำงานส่วนของ Single Sign-on ใน Globus ได้เพิ่มส่วนขยายลงบน GSI ซึ่งส่วนที่เพิ่มลงไปมีพื้นฐานมาจาก Generic Security Service API ซึ่งมาตรฐานของ API นำเสนอโดย IETF

บทที่ 4
เครื่องมือทางคลัสเตอร์
TORQUE-2.0.0p7 Scheduler

4. ภาพรวมของ TORQUE

4.1. การลงระบบ TORQUE

หลังจากทำการดาวน์โหลดไฟล์ของ TORQUE เพื่อใช้ลงระบบให้ทำการแตกไฟล์ออกจากแพ็คเกจแล้วทำการสร้างไฟล์นั้นบนเครื่องที่จะทำเป็น TORQUE server ซึ่งเป็นเครื่องที่จะทำการติดตามและควบคุมโหนดประมวลผลทุกโหนดโดยทำการรัน pbs_server daemon โดยมีตัวอย่างดังนี้

```
> tar -xzvf torqueXXX.tar.gz
> cd torqueXXX
> ./configure
> make
> make install
```

สามารถทำการสร้างตัวแปรสำหรับ PATH environment ใหม่ได้ซึ่งโดยปกติแล้วหลังจากการลงระบบแฟ้มข้อมูลหลักจะอยู่ที่ /usr/local/bin และ /usr/local/sbin ในที่นี้ข้อมูลที่เป็นฐานของ TORQUE จะอยู่ที่ /usr/spool/PBS ในส่วนคลัสเตอร์ของ TORQUE จะประกอบไปด้วย 1 เฮดโหนดและโหนดประมวลผลจำนวนมาก เฮดโหนดจะทำหน้าที่สั่งให้ pbs_server daemon ทำงานและที่โหนดประมวลผลทั้งหมดจะสั่งให้ pbs_mom daemon ทำงาน คำสั่งของฝั่ง clients ที่ใช้สำหรับส่งงานและจัดการงานนั้น สามารถนำไปลงใช้งานได้บนทุกเครื่องรวมไปถึงเครื่องที่ไม่ได้ลงทั้ง pbs_server และ pbs_mom daemon ในส่วนของเฮดโหนดนั้นจะมีการรัน scheduler daemon ด้วย ซึ่งการทำงานมีขั้นตอน คือ scheduler จะทำการติดต่อกับ pbs_server เพื่อทำการตัดสินใจในการเลือกใช้ policy เพื่อเลือกทรัพยากรหรือโหนดที่เหมาะสมให้กับงาน ส่วนของแพ็คเกจของ TORQUE จะมี scheduler FIFO แบบพื้นฐานและมีโค้ดที่จะนำไปใช้สร้าง scheduler ระดับสูงๆขึ้นไป

ส่วนของผู้ใช้งานจะทำการส่งงานไปที่ pbs_server โดยใช้คำสั่ง qsub เมื่อ pbs_server ได้รับงานใหม่เข้ามามันจะทำการแจ้งให้ scheduler ทราบ จากนั้น scheduler จะทำการหาโหนดเพื่อทำการประมวลให้กับงานโดยจะส่งคำสั่งและรายการโหนดไฟล์กลับมาที่ pbs_server จากนั้น pbs_server จะทำการส่งงานไปที่โหนดแรกที่อยู่ในรายการโหนดไฟล์เข้าทำการประมวลผล

4.2. การตั้งค่าขั้นพื้นฐาน

TORQUE จะมีขั้นตอนเริ่มแรกเพื่อให้มันทำงานอยู่ 3 ขั้นตอน ขั้นตอนแรก server จะต้องทำการเริ่มต้นระบบและปรับแต่ง ถัดไปคือต้องกำหนดโหนดประมวลผลและสุดท้ายแต่ละโหนดประมวลผลต้องรู้ว่า pbs_server อยู่ตำแหน่งใด

4.2.1. การตั้งค่าเบื้องต้นบนเครื่อง server

ครั้งแรกที่ pbs_server ทำงานจะต้องสั่งให้มันทำงานด้วย –t create เพื่อเป็นการเริ่มต้นฐานข้อมูลของ server ซึ่งเป็นที่ที่เก็บค่าเกี่ยวกับการติดตั้งเกือบทั้งหมด จากนั้นจะทำการรันคำสั่ง qmgr ซึ่งจะเข้าสู่ shell prompt เพื่อการตั้งค่า pbs_server โดยการตั้งค่าอันแรก ควรเป็นการตั้งค่าตัวปฏิบัติการ ซึ่งปกติจะเป็นที่ root@headnode จากนั้นจึงสร้างคิวของงานตามด้วยการสั่งให้ scheduler ทำงาน ถ้าต้องการดูรายการการตั้งค่าที่เราใส่เข้าไปให้ใช้คำสั่ง print server ซึ่งทั้งนี้จะมีวิธีการใช้ qmgr ช่วยเหลือแบบ online โดยใช้คำสั่ง help ตัวอย่างการตั้งค่า ดังต่อไปนี้

```
set server operators = root@headnode
set server operators += username@headnode
create queue batch
set queue batch queue_type = Execution
set queue batch started = True
set queue batch enabled = True
set server default_queue = batch
set server resources_default.nodes = 1
set server scheduling = True
```

ถ้าเกิดปัญหาในการตั้งค่าให้ตรวจดูให้แน่ใจว่า ไฟล์ประมวลผลของ TORQUE ได้ถูกประมวลทั้งหมดเรียบร้อยแล้วและไฟล์ประมวลผลเหล่านั้นอยู่ใน Path ที่ถูกต้อง

วิธีอื่นนอกเหนือจากนี้ในการตั้งค่า TORQUE server ทำโดยการรันคำสั่ง torque.setup ซึ่งหลังจากที่ลงระบบของ TORQUE เรียบร้อยแล้ว ให้ตั้งค่า pbs_server daemon โดยรันคำสั่ง torque.setup <USER> package โดย package นั้นจะมี Distribution source code และ <USER> คือชื่อผู้ใช้ที่จะทำหน้าที่เป็นผู้ดูแลระบบ TORQUE จาก script ด้านบนนั้นจะเป็นการตั้งค่า batch queue แบบพื้นฐาน

4.2.2. การกำหนดโหนดประมวลผล

การให้ pbs_server ติดต่อกับ mom แต่ละตัวนั้น มันต้องรู้ว่าเครื่องใดที่มันจะทำการติดต่อ ดังนั้น แต่ละโหนดที่เป็นสมาชิกของ batch system จะต้องใส่ชื่อของมันลงในโหนดไฟล์

ด้วย ซึ่งไฟล์นี้อยู่ที่ path $TORQUE/_HOME/server_priv/nodes ซึ่งส่วนใหญ่จะใส่ชื่อของโหนดดังตัวอย่างต่อไปนี้

server_priv/nodes

node001

node002

node003

node004

ถ้าโหนดประมวลมีหน่วยประมวลมากกว่า 1 หน่วย เราสามารถกำหนดจำนวนของหน่วยประมวลผลลงไปได้ ดังตัวอย่าง

server_priv/nodes

node001 np=2

node002 np=4

...

4.2.3. การตั้งค่า TORQUE บนแต่ละโหนดประมวลผล

ถ้าใช้แพ็คเกจสำเร็จรูปของ TORQUE ในหัวข้อที่ได้กล่าวไปแล้วข้างต้น ขั้นตอนนี้จะข้ามไปได้เลย แต่ถ้าไม่ได้ใช้ให้ตั้งค่า mom โดยใส่ชื่อของเฮดโหนดลงไปที่ไฟล์ $TORQUE_HOME/server_name จากนั้นเราสามารถตั้งค่าเพิ่มเติมในส่วนที่เราต้องการลงไปที่ไฟล์ $TORQUE_HOME/mom_priv/config บนแต่ละโหนดประมวลผลดังตัวอย่าง ซึ่งการตั้งค่าในส่วนนี้เราจะกล่าวในหัวข้อถัดๆไป

mom_priv/config

$pbsserver headnode # note: hostname running pbs_server

$logevent 255 # bitmap of which events to log

หลังจากไฟล์นี้อยู่ในโหนดประมวลผลทุกโหนดแล้วให้เริ่มการทำงานของ pbs_mom บนทุกโหนด

4.2.4. การตั้งค่าครั้งสุดท้าย

หลังการตั้งค่าที่ผ่านมาเสร็จหมดแล้วให้เริ่มการทำงานของ pbs_server ใหม่ โดยใช้คำสั่งดังนี้

qterm -t quick

pbs_server

หลังจากนั้นตรวจสอบว่าแต่ละโหนดพร้อมรับงานหรือไม่โดยใช้คำสั่ง pbsnodes –a เพื่อดูรายการโหนดทั้งหมดว่ามีสถานะเป็นอย่างไร

4.3. การตั้งค่าระดับสูง

4.3.1. ตัวเลือกของการตั้งค่าทั้งหมด

ตัวเลือก	คำอธิบาย	ค่าเริ่มต้น
--enable-docs	สร้างเอกสารของ PBS	Enabled
--enable-server	สร้าง/ไม่สร้าง server	Enabled
--enable-mom	สร้าง/ไม่สร้าง mom	Enabled
--enable-clients	สร้าง/ไม่สร้าง client	Enabled
--with-tcl=DIR_PREFIX	ใช้เมื่อต้องการการสนับสนุนของ tcl ถ้าคอมไพล์ด้วย tclx จะใช้แค่คำสั่ง –with-tcl ถ้าไลบรารี่ tcl อยู่คนละที่กับไลบรารี่ tclx DIR_PREFIX ควรเป็น /usr/local	Without
--with-tclx=DIR_PREFIX	ใช้เมื่อต้องการการสนับสนุนของ tclx มีประโยชน์เมื่อ tclx และ tcl อยู่คนละที่ ถ้าไม่กำหนดแฟ้ม PREFIX ส่วนของไลบรารี่ และ include ต้องติดตั้งลงโดยค้นหาอัตโนมัติโดย system tools	Without
--enable-gui	สร้าง xpbs GUI(Graphic User Interface)	Disabled
--set-cc=ccprog	ตั้งค่าคอมไพเลอร์เพื่อนำไปใช้ ซึ่งเป็นการตั้งค่าของตัวแปร cc และจะ override cc ทุกตัวที่อยู่ในสิ่งแวดล้อมของระบบจะมีอยู่ 2 ค่าคือ cc และ gcc ถ้าใส่ “--set-cc” อย่างเดียวค่าจะตั้งเป็น cc	gcc or cc
--set-cflags=flags	ตั้งค่า flag ของคอมไพเลอร์ คือ ตัวแปร CFLAGS ถ้าใส่ “--set-flag” CFLAGS จะตั้งค่าเป็น “” โดยปกติแล้วการตั้งค่า flags จะตั้งจากชื่อที่เกี่ยวข้องในงานซึ่งเป็นๆได้หลากหลาย	Varies
--enable-debug	เปิด DEBUG flag	Disabled
--set-tmpdir=DIR	ตั้งค่าของแฟ้ม tmp ที่ pbs_mom ซึ่งเป็นส่วนประกอบหนึ่งของ Clay-specific	/tmp
--set-server-home=DIR	ตั้งค่าของแฟ้ม /home/spool บน server ให้ pbs ซึ่งปกติจะใช้ /var/spool/torque	/usr/spool/PBS
--set-server-name-	ใช้ตั้งชื่อไฟล์ที่เก็บชื่อของ server	$PBS_SERVER_

file=FILE		HOME/server_name
--set-default-server=HOSTNAME	ใช้ตั้งชื่อของคอมพิวเตอร์ที่ client จะทำการ access เข้ามาเมื่อไม่มีชื่อเครื่องกำหนดเป็นส่วนหนึ่งของชื่อ queue และไม่มีไฟล์ server_name ห้ามตั้งเป็น 'localhost'	ชื่อของเครื่องที่กำลังสร้าง
--set-environ=PATH	ตั้งค่า path ของตัวแปรสำหรับ daemon	$PBS_SERVER_HOME/pbs_environment
--enable-plock-daemons	ทำให้ daemon ล็อคตัวเองลงไปใน logical memory ถ้าตั้งค่าเป็น 1 สำหรับ pbs_server, 2 สำหรับ pbs_scheduler, 4 สำหรับ pbs_mom ถ้าตั้งเป็น 7 จะทำทั้ง 3 ตัว (ไม่สามารถใช้ใน linux ได้)	0
--enable-syslog	ใช้เพื่อทำให้ syslog ที่แจ้ง error มาใช้งาน	Enabled
--set-sched=TYPE	ตั้งค่าชนิดของ scheduler ถ้า TYPE เป็น "C" scheduler จะเขียนด้วยภาษา C "tcl" server จะใช้ TCL based Scheduler "basl" server จะใช้ rule based Scheduler "no" จะไม่มีการใช้ scheduler	"C"
--set-sched-code=PATH	ตั้งค่าชื่อของไฟล์หรือแฟ้มข้อมูลที่มี code สำหรับ scheduler อยู่ซึ่งใช้กับ basl และ C scheduler ซึ่งสำหรับ C scheduler นั้นจะเป็นชื่อของแฟ้มข้อมูลส่าน basl scheduler จะเป็นไฟล์ซึ่งชื่อไฟล์จะต่อท้ายด้วย .basl	"fifo"
--set-mansuffix=CHAR	ตั้งค่าตัวอักษรต่อหลังของ manual page	"B"
--set-tclatrsep=CHAR	ตั้งค่าตัวอักษรคุณลักษณะของ tcl separator	"."
--set-qstatrc-file=FILE	ตั้งชื่อของไฟล์ที่ qstat จะใช้งาน ถ้าไม่มีไฟล์ ".qstatrc" ในแฟ้มข้อมูล	$PBS_SERVER_HOME/qstatrc

--with-scp	ใช้โปรแกรม scp เพื่อส่งไฟล์	Without
--enable-shell-pipe	ทำให้ชื่อของ script งานผ่านทาง pipe ไปที่ shell แต่ถ้าไม่ตั้งจะต้องใส่ชื่อ script งานเป็น input	enabled
--enable-rpp	ใช้ RPP/UDP เป็นโปรโตคอลให้ทรัพยากร query ไปที่ mom ถ้าไม่ใช้งานจะใช้โปรโตคอล TCP แทน	Enabled
--enable-sp2	ตั้งค่าเพื่อแจ้ง PBS ว่าจะสร้างสำหรับใช้งาน sp2	Disabled
--enable-tcl-qstat	ตั้งค่าเพื่อทำให้คำสั่ง qstat สามารถวิเคราะห์ tcl ได้ใช้เมื่อมี –with-tcl(x) อยู่แล้ว	Disabled
--enable-array	ตั้งค่าภายใต้ IRIX ทำให้สามารถใช้ร่วมกับ SGI Origin 2000 แบบขนาน ปกติจะตรวจหาโดยอัตโนมัติจากไฟล์ /etc/config/array	Auto
--enable-nodemask	ตั้งค่านี้เพื่อให้เป็น scheduling แบบ nodemask-based บน Origin 2000	Auto
--enable-pemask	ตั้งค่าเพื่อให้เป็น scheduling แบบ pemask-based บน Cray T3e	Auto
--enable-srfs	บน Cray คุณสามารถเปิดใช้งานเพื่อสนับสนุน SRFS ได้	Disabled
--enable-depend-cache	ทำให้สามารถ cache ข้อมูลระหว่างการทำงานได้ ซึ่งอาจพังถ้าผู้ใช้เปลี่ยนค่าที่ตั้งไว้ ในการทำงานใหม่แต่เป็นการทำให้ เร็วขึ้น ประหยัดเวลา	Disabled
--disable-filesync	ทำให้การป้องกันไฟล์ระบบไม่ทำงานจนกว่าข้อมูลจะถูกเขียนลง disk ซึ่งการสั่งให้ทำงานจะทำให้ช้าลง	Disabled

4.3.2. การตั้งค่าที่ server

4.3.2.1. ภาพรวมของการตั้งค่าที่ server

มีขั้นตอนหลายอย่างที่ต้องทำให้เสร็จและแน่ใจว่า server และโหนดต่างๆสามารถติดต่อกันได้โดยตรง ซึ่งขั้นตอนการตั้งค่าบางขั้นตอนนี้ต้องทำใน qmgr โดยตรง ส่วนการตั้งค่าอื่นๆจะจัดการโดยใช้โหนดไฟล์ใน pbs_server DNS ไฟล์ เช่น /etc/hosts และไฟล์ /etc/hosts.equiv

4.3.2.2. การตั้งค่าเกี่ยวกับชื่อของ service

แต่ละโหนดรวมถึง server ต้องสามารถวิเคราะห์ชื่อของทุกโหนดๆได้ว่าเป็นชื่อของโหนดที่มันกำลังจะทำการติดต่อ ซึ่งสิ่งเหล่านี้ทำได้โดยใช้ /etc/hosts, DNS, NIS หรือวิธีอื่นๆ ซึ่งแบบ /etc/hosts เป็นวิธีที่ใช้ไฟล์ร่วมกันข้ามระบบได้มากกว่ารูปแบบอื่นๆ

วิธีพื้นฐานที่ใช้ในการตรวจสอบการตั้งค่าของชื่อ service คือการตรวจสอบว่า server กับโหนดต่างๆสามารถ ping หากันได้

4.3.2.3. การตั้งค่าโฮสท์ที่ใช้ส่งงาน

- ใช้การตรวจสอบสิทธิ์ RCmd

เมื่องานสามารถที่จะถูกส่งมาจากโฮสท์ที่หลากหลายต่างที่กัน ดังนั้น โฮสท์เหล่านี้ควรได้รับการรับรองด้วยคำสั่ง R*(เช่น rsh,rcp,etc) ซึ่งสามารถทำอย่างนั้นได้โดยการเพิ่มโฮสท์ลงไปในไฟล์ /etc/host.equiv ที่อยู่ที่เครื่องประมวลผลหรือใช้คำสั่ง R* อื่นๆเพื่อใช้ทำการตรวจสอบระดับสิทธิ์ การตั้งค่าไฟล์รูปแบบนี้ส่วนใหญ่เป็นวิธีพื้นฐานโดยใส่ข้อความบรรทัดต่างๆลงไนไฟล์ /etc/hosts.equiv ดังตัวอย่าง

hosts.equiv

#[+ | -] [hostname] [username]

mynode.myorganization.com

.....

-การใช้พารามิเตอร์ของ "submit_hosts" server

โฮสท์ที่ได้รับความน่าเชื่อถือ จะสามารถกำหนดการเข้าถึงโดยตรงโดยไม่ต้องใช้การตรวจสอบสิทธิ์ RCmd โดยการตั้งพารามิเตอร์ของโฮสท์ส่งงานที่เป็น server ผ่านทาง qmgr ดังตัวอย่าง

qmgr

> qmgr -c 'set server submit_hosts = login1'

> qmgr -c 'set server submit_hosts += login2'

> qmgr -c 'set server submit_hosts += login3'

ถ้าต้องการให้โหนดประมวลผลทั้งหมดให้เป็นโฮสท์ส่งงาน สามารถทำได้โดยตั้งค่าพารามิเตอร์ allow_node_submit ให้เป็น true

4.3.2.4. การตั้งค่า TORQUE บน Multi-hand server

ถ้า pbs_server daemon รันในโฮสท์ที่มีส่วนติดต่อเครือข่ายมากกว่า 1 อินเทอร์เฟส อินเทอร์เฟสที่ใช้จะสามารถตั้งค่าภายนอกโดยใช้แอททริบิวท์ของ server_name server ใน qmgr ได้ ซึ่งใน TORQUE 2.0 นี้เราจะใช้พารามิเตอร์ SERVERHOST ในไฟล์ torque.cfg

4.3.2.5. การกำหนดผู้ดูแลระบบที่อื่น ๆนอกเหนือจาก root

ตามปกติ root เท่านั้นที่จะสามารถทำการเริ่มต้น ติดตั้งค่าและจัดการ pbs_server daemon ซึ่งผู้ใช้อื่นนอกเหนือจากนี้ ถ้าเราต้องเพิ่มเข้าไปเพื่อจัดการกับพารามิเตอร์และระบบปฏิบัติการจะต้องตั้งค่าโดยทำคำสั่ง qmgr ดังตัวอย่าง

```
> qmgr
Qmgr: set server managers += josh@*.fsc.com
Qmgr: set server operators += josh@*.fsc.com
```

4.3.2.6. การทดสอบการตั้งค่า TORQUE

pbs_server daemon จะเริ่มต้นทำงานบน TORQUE server เมื่อไฟล์ torque.setup ถูกประมวลผลหรือเริ่มทำงานเมื่อมันถูกตั้งค่าแบบ manual ซึ่งจะต้องสั่งให้มันเริ่มทำงานใหม่ก่อนจึงจะสามารถทำงานตามการตั้งค่าใหม่ที่เปลี่ยนแปลงไว้ได้ จากตัวอย่างต่อไปนี้

```
# shutdown server
> qterm -t quick

# start server
> pbs_server

# verify all queues are properly configured
> qstat -q

# view additional server configuration
> qmgr -c 'p s'

# verify all nodes are correctly reporting
> pbsnodes -a

# submit a basic job
> echo "sleep 30" | qsub

# verify jobs display
> qstat
```

จะเห็นว่างานจากตัวอย่างนี้ควรอยู่ในสภาวะ 'Q' และจะไม่ทำงานเพราะยังไม่ได้เริ่มการทำงานของ scheduler ด้วยคำสั่ง pbs_sched

4.4. การส่งงานและการจัดการงาน

4.4.1. การส่งงาน

การส่งงานจะทำได้โดยใช้คำสั่ง qsub คำสั่งนี้มีอาร์กิวเมนต์จำนวนหนึ่ง ซึ่งต่อไปจะเอาคำสั่งนี้มารวมกันเป็น PBS batch ไฟล์ ซึ่งอาจกำหนดผ่านทางชื่อไฟล์บนคำสั่ง qsub หรืออาจจะกำหนดผ่านทาง STDIN จะมีหัวข้อที่ต้องสังเกต ดังนี้

- PBS batch script ไม่จำเป็นที่จะต้องนำไปประมวลผล
- PBS batch script อาจจะทำการ pipe('|') ลงคำสั่ง qsub เช่น 'cat pbs.cmd | qsub'
- ในกรณีที่การส่งงานเป็นแบบขนาน script ของคำสั่ง PBS จะประมวลผลที่โหนดประมวลผลที่หาได้โหนดแรกโหนดเดียวเท่านั้น(ถ้าต้องการรันบนหลายๆโหนดจะต้องใช้คำสั่ง pbsdsh ในการรัน)
- Batch script จะถูกประมวลผลจากแฟ้มเอกสาร Home ของผู้ใช้ในทุกกรณี(สามารถเปลี่ยนแฟ้มเอกสารในการส่งงานได้โดยการตั้งค่าตัวแปรแวดล้อม $PBS_O_WORKDIR)
- Batch script จะถูกประมวลผลโดยใช้ตัวแปรปกติทั้งหมด ถ้าต้องการกำหนดสิ่งใดเพิ่มเข้าไปสำหรับการส่งงานจะต้องกำหนด flag '-V' หรือ –v ไว้ด้วย
- คำสั่ง pbsdsh จะไม่มีการรับค่าอาร์กิวเมนต์ใน batch script

*หมายเหตุ โดยปกติการส่งงานจะทำได้ที่เครื่องที่เป็น server ที่มี pbs_server ทำงานอยู่เท่านั้น

4.4.2. การร้องขอทรัพยากร

ทรัพยากรที่หลากหลายสามารถร้องขอได้ในเวลาที่มีการส่งงาน งานงานหนึ่งสามารถร้องขอทรัพยากรแบบชี้เฉพาะตามความต้องการของงานนั้นๆได้ ทั้งเจาะจงไปที่การขอโหนดใดโหนดหนึ่งหรือเจาะจงลงไปที่แอททริบิวต์ใดๆแอททริบิวต์หนึ่งหรือ เจาะจงทั้งแอททริบิวต์ เจาะจงทั้งที่โหนด ก็ได้ ทรัพยากรจะมีที่มาอยู่ 2 ลักษณะ คือ ทรัพยากรที่ TORQUE มีมาให้และทรัพยากรที่มาจาก scheduler ภายนอก โดยทรัพยากรที่ TORQUE มีมาให้มีดังต่อไปนี้

ตารางที่ 4.1

ทรัพยากร	รูปแบบ	รายละเอียด
arch	string	กำหนดว่าต้องการโครงสร้างของผู้ดูแลระบบแบบใดซึ่งปกติเป็นสิ่งที่ PBS_MACH string ทำการตั้งค่าเข้าไปที่ local.mk
cput	วินาที, หรือ [[HH:]MM:]SS	เวลาทั้งหมดที่ใช้ไปเพื่อการ

		ประมวลผลงานทั้งหมด
file	size*	พื้นที่ว่างในดิสก์ทั้งหมดที่งานร้องขอ
host	string	ชื่อของโฮสท์ที่งานจะทำการรัน ทรัพยากรประเภทนี้ใช้สำหรับการกำหนด scheduling policy
mem	size*	จำนวน physical memory ทั้งหมดที่งานร้องขอ
nice	integer	จำนวนเต็มระหว่าง -20 ถึง 19 ซึ่งใช้ปรับค่าความสำคัญของการประมวลผลว่า process ใดสำคัญมากกว่าหรือน้อยกว่ากัน โดยที่ -20 คือ ค่า priority สูงสุด และ 19 คือค่า priority ที่ต่ำที่สุด
nodes	{<node_count> | <hostname>} [:ppn=<ppn>][:<property>[:<property>]...] [+ ...]	จำนวนและชนิดของโหนดที่สำรองไว้สำหรับงาน ค่าที่ใส่คือ "node_spec[+node_spec...]" ซึ่งแต่ละ node_spec คือ จำนวนโหนดที่ต้องการประกาศชนิดที่ได้ทำการตั้งไว้ใน node_spec หรือคุณสมบัติที่ต้องการอื่นๆของโหนด จำนวนชื่อ ซึ่งจะคั่นด้วยเครื่องหมาย ':' ถ้าไม่มีการกำหนดจำนวนจะตั้งไว้ที่ 1 และ <hostname> คือ ชื่อของโหนด โดยคุณสมบัติของโหนดมีดังนี้ 1. ppn # กำหนดจำนวนหน่วยประมวลผลต่อโหนดที่ต้องการ 1

		โหนด ซึ่งค่าเริ่มต้นจะเป็น 1 2. property จะใส่ค่า string โดยผู้ดูแลระบบเพื่อกำหนดความสามารถของโหนด
opsys	string	กำหนดระบบปฏิบัติการให้เป็นเหมือนที่กำหนดไว้ในไฟล์ mom config
other	string	อนุญาตให้ผู้ใช้สามารถกำหนดข้อมูลเฉพาะอย่างเข้าไปที่ไซต์ได้ ทรัพยากรประเภทนี้ใช้สำหรับกำหนด scheduling policy ของไซต์ การตั้งค่าจึงขึ้นอยู่กับความต้องการของไซต์นั้นๆ
pcput	วินาที, [[HH:]MM:]SS	เวลามากสุดที่หน่วยประมวลผลใช้ในการประมวลผล process เดี่ยวของงาน
pmem	size*	จำนวน physical memory เต็มที่ที่ใช้ใน process เดี่ยวของงาน
vmem	size*	จำนวน virtual memory เต็มที่ที่ใช้ใน process เดี่ยวของงาน
walltime	วินาที, [[HH:]MM:]SS	เวลาเต็มที่แบบ real-time ในระหว่างที่งานอยู่ใน running state

*หมายเหตุ size หมายถึง รูปแบบที่บอกปริมาณในหน่วยของ bytes หรือ words

Suffix		Multiplier
b	w	1
kb	kw	1024
mb	mw	1,048,576
gb	gw	1,073,741,824
tb	tw	1,099,511,627,776

ตัวอย่างการใช้งาน 1 (-l nodes)

การใช้งาน	**รายละเอียด**
qsub -l > qsub -l nodes=12	ร้องขอจำนวนโหนด 12 โหนดที่เป็นชนิดใดก็ได้
qsub -l > qsub -l nodes=2:server+14	ร้องขอ 2 โหนด server และโหนดอื่นๆ อีก 14 โหนด เป็นการกำหนด node_specs 2 รูปแบบ คือ "2:server" และ "14"
qsub -l > qsub -l nodes=server:hippi+10:noserver+3:bigmem:hippi	ร้องขอ 1 โหนด server และ server นั้นมี interface ในรูปแบบ "hippi" และ 10 โหนดที่ไม่ใช่ server และ อีก 3 โหนดที่มีปริมาณหน่วยความจำขนาดใหญ่และมี interface รูปแบบ "hippi"
qsub -l > qsub -l nodes=b2005+b1803+b1813	ร้องขอโหนดเฉพาะ 3 โหนดที่กำหนดโดยใส่ชื่อของโฮสท์
qsub -l > qsub -l nodes=4:ppn=2	ร้องขอตัวประมวลผล 2 ตัวบนแต่ละ 4 โหนด
qsub -l > qsub -l nodes=1:ppn=4	ร้องขอโหนดเดียวที่มีตัวประมวลผล 4 ตัว
qsub -l > qsub -l nodes=2:blue:ppn=2+red:ppn=3+b1014	ร้องขอตัวประมวลผล 2 ตัวบนแต่ละ2 โหนด blue และ 3 โหนด red ที่มีตัวประมวลผล 3 ตัว และ โหนดประมวลผลที่มีชื่อโฮสท์ว่า "b1014"

ตัวอย่างการใช้งาน 2

qsub -l mem=200mb /home/user/script.sh

งานนี้ต้องการโหนดที่มีหน่วยความจำเหลือ 200 MB

ตัวอย่างการใช้งาน 3

> qsub -l nodes=node01,mem=200mb /home/user/script.sh

งานนี้จะรอจนกว่า node01 **จะมีหน่วยความจำเหลือ** 200 MB **แล้วส่งงาน**

4.4.3. **การติดตามสถานะงาน** (Monitoring Jobs)

TORQUE มีความสามารถที่ทำให้ผู้ใช้หรือผู้ดูแลระบบสามารถติดตามสถานะของงานที่ได้ทำการส่งเข้าประมวลแล้วด้วยคำสั่ง qstat โดยผลลัพธ์ของการรันคำสั่งนี้เป็นดังตัวอย่างต่อไปนี้

```
> qstat
Job id          Name            User            Time Use S Queue
---------- ---------- ---------- ---------- - --------
4807            scatter         user01          12:56:34 R batch
...
```

4.4.4. **การยกเลิกงาน** (Canceling Jobs)

TORQUE มีคำสั่ง qdel เพื่อให้ผู้ใช้หรือผู้ดูแลระบบสามารถทำการยกเลิกงานที่ทำการส่งไปแล้ว โดยจะส่งสัญญาณไปยกเลิก process ที่กำลังทำการประมวลผล เมื่อสคริปท์ที่อยู่ระดับบนสุดยกเลิก งานจึงจะยกเลิกตาม

ถ้างานนั้นๆถูกยกเลิกโดยผู้มีหน้าที่จัดการ จะมีการแจ้งอีเมล์ไปยังผู้ใช้โดย ผู้จัดการสามารถใส่ข้อความเป็นหมายเหตุไปกับอีเมล์ได้โดยใส่ตัวเลือก –m

ตัวอย่างการใช้งาน qdel

```
$ qstat
Job id          Name            User            Time Use S Queue
---------- ---------- ---------- ---------- - -----
4807            scatter         user01          12:56:34 R batch
...
$ qdel -m "hey! Stop abusing the NFS servers" 4807
$
```

4.4.5. การจองงาน (Job Preemption)

TORQUE มีความสามารถในการจองงานโดยอนุญาตให้ผู้ใช้ที่มีสิทธิ์ถูกต้อง สามารถทำการหยุดพักงาน หรือ ทำให้งานที่พักไว้เริ่มทำงานต่อได้ ความสามารถนี้เราทำได้ 2 วิธี หนึ่งในนั้นถ้าโหนดสามารถจองงานในระดับของระบบปฏิบัติการ TORQUE จะตรวจจับได้ระหว่างการตั้งค่า process แล้วทำการ enable การจองให้เกิดขึ้น ซึ่งนอกเหนือจากวิธีนี้จะทำโดย mom จะถูกตั้งค่าให้มี script ที่เอาไว้ทำเช็คพ้อยท์ ซึ่งเพื่อสนับสนุนการใช้เช็คพ้อยท์นี้ งานนั้นๆจะต้องรู้ว่าตัวเองจะกลับสู่การทำงานต่ออย่างไรจากจุดเช็คพ้อยท์นั้นหลังเกิดการจอง

4.4.6. การตั้งค่าเช็คพ้อยท์สคริปท์ที่ mom

การตั้งค่าเช็คพ้อยท์สคริปท์ที่ mom นี้ $checkpoint_script พารามิเตอร์ต้องตั้งค่าลงในไฟล์ตั้งค่าของ mom ซึ่งอยู่ที่ path $TORQUEHOME/mom_priv/config โดยมีรูปแบบดังตัวอย่าง

mom_priv/config

```
$pbsserver      node06
$logevent       255
$restricted     *.mycluster.org
$checkpoint_script  /opt/moab/tools/mom-checkpoint.sh
```

อย่างที่สองที่ต้องทำ คือ เปลี่ยนค่าของ MOM_CHECKPOINT ให้มีค่าเป็น 1 โดยไปเปลี่ยนได้ที่ .../src/include/pbs_config.h โดยมีตัวอย่างดังนี้

.../src/include/pbs_config.h

```
#define MOM_CHECKPOINT 1
```

4.4.7. งานที่เสร็จสมบูรณ์

TORQUE มีความสามารถในการแจ้งสถานะของงานที่ประมวลผลเสร็จแล้ว เพื่อการตั้งค่าในช่วงเวลาหลังงานเสร็จสมบูรณ์ ความสามารถนี้นำมาใช้ได้โดยการตั้งค่าแอททริบิวต์ keep_completed บนคิวที่ประมวลผลงาน ด้วยการตั้งค่าจำนวนวินาทีที่งานจะโดนดึงไว้ในคิวประมวลผล งานที่เสร็จสิ้นแล้วจะแสดงสถานะด้วยตัวอักษร 'C' และแสดงสถานะ 'exit_status' ที่แอททริบิวต์ของงาน

4.5. การจัดการโหนด

4.5.1. Adding Nodes

Torque สามารถที่จะเพิ่มหรือลดโหนดได้แบบไดนามิกโดยใช้คำสั่ง qmgr หรืออาจไปแก้ไขที่ไฟล์ *$TORQUE_HOME/server_priv/nodes*

Torque สามารถที่จะเพิ่มโหนดได้แบบไดนามิกโดยใช้คำสั่ง qmgr ดังตัวอย่างที่จะเป็นการเพิ่มโหนดที่ชื่อว่า node003

> qmgr -c "create node node003"

ซึ่งชื่อของโหนดจะไปปรากฏที่ไฟล์ *$TORQUE_HOME/server_priv/nodes*

Node003

เราสามารที่จะลบโหนดออกได้โดยใช้คำสั่ง

> qmgr -c "remove node node003"

แต่ปกติแล้ว แอดมินมักจะเปลี่ยนแปลงสถานะของโหนดมากกว่าที่จะลบโหนดนั้นทิ้ง ถ้าเป็นTorque 2.0 แล้ว แนะนำว่าถ้าเปลี่ยนแปลงโหนดเสร็จแล้ว ให้ทำการรีสตาร์ต pbs_server ด้วย หรืออาจเปลี่ยนแปลงไฟล์ก่อนแล้วค่อยรีสตาร์ต pbs_server

4.5.2. Nodes Properties

Torque สามารถเชื่อมโยงคุณสมบัติกับโหนดต่างๆเพื่อช่วยเหลือในการระบุกลุ่มของโหนดได้ คุณสมบัติต่างๆสามารถถูกเพิ่มไปให้โหนดแต่ละตัวได้ ตัวอย่างเช่น กลุ่มของโหนดที่มีการเชื่อต่อเครือข่ายที่รวดเร็ว อาจจะมีคุณสมบัติที่เรียกว่า "ib" Torque สามารถตั้งค่าคุณสามบัติเปลี่ยนแปลง หรือลบคุณสมบัติของโหนดต่างๆได้โดยใช้คำสั่ง qmgr หรืออาจไปแก้ได้ที่ไฟล์ nodes

4.5.2.1. Run-Time Node Changes

Torque สามารถที่จะเปลี่ยนแปลงคุณสมบัติของโหนดต่างๆได้โดยการใช้คำสั่ง qmgr เช่น ถ้าต้องการทำให้ node001 มีคุณสมบัติเป็น "bigmem"และ"dualcore" ต้องพิมพ์คำสั่งดังนี้

> qmgr -c "set node node001 properties = bigmem"

> qmgr -c "set node node001 properties += dualcore"

หากต้องการที่จะยกเลิกคุณสมบัติ จำใช้โอเปอเรเตอร์ -=

4.5.2.2. Manual Node Changes

คุณสมบัติของแต่ละโหนดนั้นจะถูกบันทึกลงไฟล์ $TORQUEHOME/server_priv/nodes โดยคุณสมบัติที่ถูกเพิ่มเติมเข้ามานั้น จะต้องเขียนข้างหลังชื่อของโหนด ตัวอย่างเช่น ถ้าต้องการกำหนดให้ "node001" มีคุณสมบัติเป็น "bigmem"และ"dualcore" และต้องการกำหนดให้ "node002" มีคุณสมบัติเป็น "bigmem"และ"matlab" ให้เขียนnode file ดังนี้

node001 bigmem dualcore

node002 np=4 bigmem matlab

ถ้าnode file มีการเปลี่ยนแปลง จะต้องทำการรีสตาร์ต pbs_server ใหม่ทุกครั้ง

4.5.2.3. Changing Node State

วิธีการที่จะป้องกันไม่ให้ส่งงานไปรันเป็นบางเครื่องนั้น เราจะเปลี่ยนสถานะของเครื่องนั้นๆให้เป็นออฟไลน์ โดยใช้คำสั่ง pbsnodes -o nodename จากนั้น เมื่อโหนดถูกเปลี่ยนสถานะเป็นออฟไลน์ scheduler จะไม่ทำการส่งงานใหม่ไปยังเครื่องนั้นๆ หากต้องการให้โหนดนั้นกลับมาใช้งานได้เหมือนเดิม เราจะใช้คำสั่ง pbsnodes –s nodename

คำสั่งอีกอันที่มีประโยชน์มาก ก็คือคำสั่ง **pbsnodes –l** ซึ่งคำสั่งนี้จะทำการแสดงรายชื่อโหนดทุกโหนดที่มีสถานะแปลกๆ เช่น down, unknown หรือ offline ซึ่งจะช่วยให้เราสามารถเช็กได้ง่ายว่าเครื่องใดบ้างที่มีปัญหา

4.5.2.4. Host Security

สำหรับระบบที่ต้องการจะติดต่อกับโหนดประมวลผล(ตัวอย่างเช่น ผู้ใช้ที่มีข้อมูลที่สำคัญ) Torque's prologue script และ epilogue script จะจัดเตรียมสื่อที่ใช้ในการทำ authentication ซึ่งถูกจัดการโดย โมดูลของ linux-PAM

หากต้องการที่จะให้เฉพาะuserที่กำลังรันงาน สามารถติดต่อกับโหนดประมวลผล จะมีวิธีการดังนี้

- คลายไฟล์(Untar) contrib./pam_authuser.tar.gz
- คอมไพล์ไฟล์ pam_authuser.c ด้วยคำสั่ง make และ make install บนโหนดประมวลผลทุกโหนด
- แก้ไขไฟล์ /etc/system-auth ตามคำอธิบายใน README.pa,_authuser บนโหนดประมวลผลทุกโหนด
- สร้าง tar ball ของสคริปต์ epilogue* และสคริปต์ prologue* แล้วทำการคลายไฟล์ (Untar) ลงในไดเร็กทอรี่ mom_priv หรืออาจจะก็อปปี้ epilogue* และ prologue* ไปยังmom_priv/. ก็ได้

สคริปต์ prologue* นั้นเป็น perl script ซึ่งจะทำการเพิ่มuserของงานไปยังไฟล์ /etc/authuser ส่วนสคริปต์ epilogue* นั้นจะทำหน้าที่ลบuserที่ปรากฏเป็นอันแรก ออกจากไฟล์ /etc/authuser ซึ่งในสคริปต์ epilogue* นั้นจะมีโค้ดที่ถูกคอมเม้นอยู่ ซึ่งหากเอาคอมเม้นออก จะทำการฆ่าโพรเซสทั้งหมดที่ถูกเป็นเจ้าของโดยuserนั้น เมื่อuserนั้นไม่มีงานอื่นรันอยู่บนโหนดเดียวกัน

Prologue* และepilogue* script นั้นจะถูกเพิ่มไปยังpam_authuser tar ball ใน Torque เวอร์ชัน 2.1

4.6. Queue Configuration

ภายใต้Torque องค์ประกอบของคิวนั้นจะสมบูรณ์ก็ต่อเมื่อใช้คำสั่ง qmgr โดยขั้นตอนแรกนั้น เราจะต้องทำการสร้างคิวขึ้นมาก่อน ซึ่งเราสามารถทำได้โดยใช้ create queue ซึ่งเป็นคำสั่งย่อยของ qmgr ตามตัวอย่าง

>qmgr –c "create queue batch queue_type=execution"

เมื่อคิวถูกสร้างขึ้นมาแล้ว ก็จะต้องมีการปรับแต่งเพื่อให้มันสามารถใช้งานได้ โดยอย่างน้อยที่สุด จะต้องมีการใช้ออพชั่น started และ enabled นอกจากนี้แล้วยังสามารถปรับแต่งโดยใช้ attributeต่างๆมาผสมกันตามข้อมูลข้างล่าง

สำหรับข้อมูลที่เป็นชนิด Boolean นั้น T, t, 1, Y และ y นั้นจะเปรียบเหมือนกับค่าที่เป็น true ส่วน F, f, 0, N และ n จะหมายถึง false

4.6.1. Queue Attributes

Parameter	Format	Description	Example
acl_groups	[-]<GROUP>[@<HOST>][+<USER>[@<HOST>]]...	ระบุข้อมูลของกลุ่มที่สามารถส่งงานหรือกลุ่มที่ไม่สามารถส่งงานไปยังคิว ถ้า acl_group_enable ถูกตั้งเป็น true, เฉพาะงานที่มีผลกับกลุ่มใน acl_groups เท่านั้น ที่คิวจะสามารถใช้ได้ ถ้าแอททริบิวของคิว acl_group_sloppy ถูกตั้ง acl_groups จะ	> qmgr -c "set queue batch acl_groups=staff" > qmgr -c "set queue batch acl_groups+=ops@h2" > qmgr -c "set queue batch acl_groups+=staff@h3" > qmgr -c "set queue batch acl_groups+=-bioasp Note: ใช้ร่วมกับ acl_group_enable

		ถูกเช็กไปยังกลุ่มที่สองของงาน	
acl_group_enable	<BOOLEAN>	ถ้า TRUE, ข้อจำกัดของ Torqueจะอนุญาตให้เฉพาะงานที่ถูกส่งมาจากกลุ่มที่ถูกรบุโดยพารามิเตอร์ของ acl_groups เท่านั้น ค่าเริ่มต้น: FALSE	> qmgr -c "set queue batch acl_group_enable=true"
acl_group_sloppy		ถ้าค่าเป็น TRUE, acl_groups จะถูกเช็กกับทุกกลุ่มที่userของงานนั้นเป็นสมาชิกอยู่ ค่าเริ่มต้น: FALSE	
acl_hosts	<HOST>[+<HOST>]...	ระบุว่าhost ใดบ้างที่จะส่งงานไปยังคิวได้	> qmgr -c "set queue batch acl_hosts=h1+h2+h3" Note: used in conjunction with acl_host_enable
acl_host_enable	<BOOLEAN>	ถ้าค่าเป็น TRUE, ข้อจำกัดของ TORQUE จะอนุญาตเฉพาะงานที่มาจากhost ที่ถูก	> qmgr -c "set queue batch acl_host_enable=true"

		ระบุใน acl_hosts เท่านั้น ค่าเริ่มต้น: FALSE	
acl_users	<USER>[@<HOST>] [+<USER>[@<HOST>]]...	ระบุรายการของ user ที่จะส่งงานไปยังคิว ถ้า acl_user_enable ถูกตั้งเป็น TRUE, เฉพาะ userที่มีชื่อในรายการเท่านั้นที่สามารถส่งงานเข้าคิวได้	> qmgr -c "set queue batch acl_users=john" > qmgr -c "set queue batch acl_users+=steve@h2" > qmgr -c "set queue batch acl_users+=stevek@h3" Note: ถูกใช้ร่วมกับ acl_user_enable
acl_user_enable	<BOOLEAN>	ถ้าค่าเป็นTRUE, เงื่อนไขของ TORQUE จะอนุญาตให้งานที่ถูกส่งมาจาก user ที่ถูกระบุโดยพารามิเตอร์ของacl_user เท่านั้น ค่าเริ่มต้น: FALSE	> qmgr -c "set queue batch acl_user_enable=true"
acl_logic_or		ถ้าตั้งค่าเป็น TRUE, user และกลุ่มของ acls จะถูกตั้งค่าลอจิกเป็น OR ซึ่งหมายความว่า acl แต่ละอัน	

		อาจจะถูกพบว่าอนุญาตให้เข้าถึงได้ ถ้ามีค่าเป็น false หรือ unset แล้วค่าลอจิกจะเป็น AND, ค่าเริ่มต้น: FALSE	
enabled	<BOOLEAN>	ระบุว่าถ้าจะยอมให้คิวรับงานใหม่ได้หรือไม่ ค่าเริ่มต้น: FALSE	> qmgr -c "set queue batch enabled=true"
keep_completed		ระบุจำนวนวินาทีที่จะให้งานค้างไว้ก่อนตอนที่ขึ้นสถานะว่า Complete ค่าเริ่มต้นเป็น 0.	
kill_delay	<INTEGER>	ระบุจำนวนวินาทีระหว่างที่ส่ง SIGTERM และ SIGKILL เพื่อให้ยกเลิกงาน. ค่าเริ่มต้น: 2 วินาที	> qmgr -c "set queue batch kill_delay=30"
max_queuable	<INTEGER>	ระบุจำนวนมากที่มากที่สุดที่อนุญาตให้อยู่ในคิวได้ในช่วงเวลาใดเวลาหนึ่ง (รวมถึงเวลาที่	> qmgr -c "set queue batch max_queuable=20"

		ว่าง รันงาน และบล็อกงาน) ค่าเริ่มต้น: ไม่จำกัด	
max_user_queuable	<INTEGER>	ระบุจำนวนงานที่มากที่สุดต่อ 1 user ที่อนุญาตให้อยู่ในคิวได้ในช่วงเวลาใดเวลาหนึ่ง (รวมถึงเวลาที่ว่าง รันงาน และบล็อกงาน) ค่าเริ่มต้น: ไม่จำกัด เฉพาะตั้งแต่เวอร์ชัน 2.1.3 ขึ้นไป	> qmgr -c "set queue batch max_user_queuable=20"
max_running	<INTEGER>	ระบุจำนวนงานที่มากที่สุดที่อนุญาตให้รันได้ในช่วงเวลาใดเวลาหนึ่ง ค่าเริ่มต้น: ไม่จำกัด	> qmgr -c "set queue batch max_running=20"
max_user_running	<INTEGER>	ระบุจำนวนงานที่มากที่สุดต่อ1 user ในคิวที่อนุญาตให้รันได้ในช่วงเวลาดเวลาหนึ่ง ค่าเริ่มต้น: ไม่จำกัด	> qmgr -c "set queue batch max_running=20"
priority	<INTEGER>	ระบุค่าpriority ให้กับคิว ค่า	> qmgr -c "set queue batch priority=20"

		เริ่มต้น: 0	
queue_type	one of **e**, **execution**, **r**, or route	**ระบุชนิดของคิว ค่านี้จะถูกตั้งให้กับคิวทุกคิว**	> qmgr -c "set queue batch queue_type=execution"
resources_available	<STRING>	**ระบุทรัพยากรที่เพิ่มขึ้นไปยังทุกงานที่อยู่ในคิว ค่าเริ่มต้น**: N/A	> qmgr -c "set queue batch resources_available.nodect=20" Note: จะต้องรีสตาร์ต pbs_server เพื่อให้สิ่งที่เราเปลี่ยนแปลงไว้มีผล จากนั้น resources_available ก็จะมีข้อกำหนดขั้นต่ำสุดของ queue.resources_available และ server.resources_available.
resources_default	<STRING>	**ระบุค่าเริ่มต้นของความต้องการทรัพยากรของงานที่ถูกส่งไปยังคิว ค่าเริ่มต้น**: N/A : ค่าเริ่มต้นอย่างน้อยต้องตั้งค่าทรัพยากรของ walltime และ nodes ด้วย	> qmgr -c "set queue batch resources_default.walltime=3600"
resources_max	<STRING>	**ระบุค่าที่มากที่สุดของทรัพยากรให้กับ**	> qmgr -c "set queue batch resources_max.nodect=16"

		งานที่ถูกส่งมายังคิว **ค่าเริ่มต้น**: N/A	
resources_min	<STRING>	**ระบุทรัพยากรขั้นต่ำให้กับงานที่ถูกส่งไปยังคิว ค่าเริ่มต้น**: N/A	> qmgr -c "set queue batch resources_min.nodect=2"
route_destinations	<queue>[@<host>] [+<queue>[@<host>]] ...	ระบุคิวเป้าหมายที่เป็นไปได้ ให้กับงานที่เกี่ยวข้องกับ routing queue. **ค่าเริ่มต้น**: N/A	> qmgr -c "set queue route route_destinations=fast" > qmgr -c "set queue route route_destinations+=slow"
started	<BOOLEAN>	**ระบุว่าจะอนุญาตให้งานในคิวถูกประมวลผลหรือไม่ ค่าเริ่มต้น**: FALSE	> qmgr -c "set queue batch started=true"

4.6.2. Example Queue Configuration

ชุดคำสั่งของqmgrต่อไปนี้ จะทำการสร้างและปรับแต่งคิวหนึ่งที่มีชื่อว่า batch

```
qmgr -c "create queue batch queue_type=execution"
qmgr -c "set queue batch started=true"
qmgr -c "set queue batch enabled=true"
qmgr -c "set queue batch resources_default.nodes=1"
qmgr -c "set queue batch resources_default.walltime=3600"
```

คิวอันนี้จะยอมรับงานใหม่ และถ้าไม่ระบุงานให้ชัดเจน มันจะกำหนดทุกงานให้ nodecount **เป็น** 1 **และ** walltime**เป็น** 1 **ชั่วโมง**

4.6.3. Setting a Default Queue

โดยปกติแล้ว งานจะต้องระบุสเป็กว่าคิวแบบไหนที่มันจะไปรัน เพื่อที่จะเปลี่ยนแปลงพฤติกรรมดังกล่าว พารามิเตอร์ของเซิฟเวอร์ที่ชื่อว่า default_queue อาจจะระบุตามตัวอย่างต่อไปนี้

qmgr –c "set server default_queue=batch"

4.6.4 Mappinga Queue to a Subset of Resources

Torque ปัจจุบันยังไม่สามารถจับคู่คิวไปยังโหนดต่างๆได้ อย่างไรก็ตาม ตัวscheduler ต่างๆอย่างเช่น Moab และ Maui สามารถที่จะทำงานฟังก์ชันดังกล่าวได้

วิธีที่ง่ายที่สุดก็คือการใช้ default_resource.neednodes บน execution queue ทั้งMauiและ Moab จะใช้ข้อมูลเหล่านี้เพื่อมันใจได้ว่างานที่อยู่ในคิวนั้นจะถูกจัดให้โดยโหนดที่ถูกระบุในattribute สำหรับตัวอย่างนั้น จะสมมติว่าเรามีบางโหนดที่ซื้อมาจากแผนกเคมี และบางโหนดถูกแจกจ่ายไปยังแผนกชีวะ

```
$TORQUE_HOME/server_priv/nodes:
node01 np=2 chem
node02 np=2 chem
node03 np=2 bio
node04 np=2 bio
qmgr:
set queue chem resources_default.neednodes=chem
set queue bio  resources_default.neednodes=bio
```

ปล. ตัวอย่างอันนี้จะไม่สามารถป้องกันคิวอันอื่นที่มากจากโหนดเหล่านั้นได้ หนึ่งในวิธีแก้ปัญหาก็คือการใช้attributeอีกอย่างนึงกับโหนดและคิวอันอื่น

4.6.5. Creating a Routing Queue

Routing queue นั้นจะนำทางงานไปยังคิวปลายทางโดยขึ้นอยู่กับattributeของงาน และข้อจำกัดของคิว Routing queue นั้นจะถูกติดตั้งโดยการสร้างคิวของ queue_type กำหนดเส้นทางโดย route_destinations โดยattributeต่างๆนั้นจะถูกตั้งตามตัวอย่างดังต่อไปนี้

```
# routing queue
create queue route
set queue route queue_type = Route
set queue route route_destinations = reg_64
set queue route route_destinations += reg_32
set queue route route_destinations += reg
```

```
set queue route enabled = True
set queue route started = True
# queue for jobs using 1-15 nodes
create queue reg
set queue reg queue_type = Execution
set queue reg resources_min.ncpus = 1
set queue reg resources_min.nodect = 1
set queue reg resources_default.ncpus = 1
set queue reg resources_default.nodes = 1
set queue reg enabled = True
set queue reg started = True
# queue for jobs using 16-31 nodes
create queue reg_32
set queue reg_32 queue_type = Execution
set queue reg_32 resources_min.ncpus = 31
set queue reg_32 resources_min.nodes = 16
set queue reg_32 resources_default.walltime = 12:00:00
set queue reg_32 enabled = True
set queue reg_32 started = True
# queue for jobs using 32+ nodes
create queue reg_64
set queue reg_64 queue_type = Execution
set queue reg_64 resources_min.ncpus = 63
set queue reg_64 resources_min.nodes = 32
set queue reg_64 resources_default.walltime = 06:00:00
set queue reg_64 enabled = True
set queue reg_64 started = True
# have all jobs go through the routing queue
set server default_queue = batch
set server resources_default.ncpus = 1
set server resources_default.walltime = 24:00:00
```

ในตัวอย่างนี้ โหนดประมวลผลจะเป็นdual processors และ ค่าwalltime เริ่มต้นจะถูกตั้งให้ขึ้นอยู่กับจำนวนโพรเซสเซอร์/จำนวนโหนดของงาน งานที่โหนด 32 (63 โพรเซสเซอร์)หรือมากกว่านั้นจะถูกตั้งwalltimeไว้ที่ 6 ชั่วโมง แต่งานที่โหนด 16-31 (31-62 โพรเซสเซอร์) จะถูกตั้งwalltimeไว้ที่ 12 ชั่วโมง และงานอื่นๆที่นอกเหนือจากนี้จะถูกตั้งwalltimeไว้ที่ 24 ชั่วโมง

ลำดับของ route_destinations ก็เป็นสิ่งสำคัญ ในการหาเส้นทางของคิวนั้น งานจะถูกส่งไปยังคิวเป้าหมายแรกที่สามารถส่งไปได้ โดยขึ้นอยู่กับค่าแอททริบิว resource_max , resource_min , acl_users และ acl_groups ในตัวอย่างข้างต้น แอททริบิวของงานที่มีโพรเซสเซอร์เดียวจะถูกตรวจสอบกับคิว reg_64 ก่อน จากนั้นจึงตรวจสอบกับคิว reg_32 แล้วสุดท้ายค่อยตรวจกับคิว reg

จากนั้นให้ระบุความต้องการรีซอร์สของคิว โดยเพิ่มจากตัวอย่างข้างบน

```
set queue reg resources_max.ncpus = 30
set queue reg resources_max.nodect = 15
set queue reg_16 resources_max.ncpus = 62
set queue reg_16 resources_max.ncpus = 31
```

เวลาของ enforcement ของเซิฟเวอร์และคิวนั้นเป็นสิ่งสำคัญมากในตัวอย่างนี้ Torque สามารถประยุกต์ค่าเริ่มต้นของเซิฟเวอร์และคิวให้แตกต่างกันได้ในโหมด job centric และโหมด queue centric สำหรับโหมด job centric นั้น Torque รอที่จะประยุกต์ค่าเริ่มต้นของเซฟเวอร์และคิวจนกว่างานที่ถูกส่งไปยังคิว execution จะเสร็จ สำหรับโหมด queue centric มันจะบังคับค่าเริ่มต้นของ server ก่อนที่มนจะถูกเอาไปใส่ใน routing queue ในแต่ละโหมด ค่าเริ่มต้นของคิวจะดีกว่าค่าเริ่มต้นของ server ปกติแล้ว Torque จะเป็นโหมด job centric โดยการตั้งให้เป็นโหมด queue centric ให้เปลี่ยนค่าของ queue_centric_limit ดังนี้

```
set server queue_centric_limits = true
```

สิ่งที่ได้จากโหมด job centric ก็คือ ถ้างานไม่ตั้งแอททริบิวไว้ ค่าเริ่มต้นของ server และ routing queue จะไม่ถูกประยุกต์เมื่อขีดจำกัดของทรัพยากรของคิวถูกตรวจสอบ เพราะฉะนั้น งานที่ต้องการใช้โหนด 32 โหนด(ไม่ใช่ ncpus = 32) จะไม่ถูกตรวจสอบขีดจำกัดกับ min_resource.ncpus เช่นเดียวกับตัวอย่างข้างต้น งานที่ไม่ได้ตั้งค่าแอททริบิว จะถูกนำไปใส่ในคิว reg_64 จนกว่าค่าเริ่มต้นของ server ncpus จะถูกประยุกต์ หลังจากงานที่ถูกส่งไปยังคิว execution

4.7. Configuring Data Management

4.7.1. การติดตั้ง SCP/RCP

การใช้ประโยชน์จาก SCP ในการจัดการกับข้อมูล TORQUE จะต้องได้รับอนุญาตในการเคลื่อนย้ายข้อมูลไปยัง node อื่นๆ ถ้าหากว่ายังไม่มีการใช้ในคลัสเตอร์ก็สามารถเปิดใช้งานได้โดยคำสั่งการทำงานด้านล่างนี้ ซึ่งการทำงานนี้เป็นการเปิดใช้แบบทิศทางเดียวสำหรับผู้ใช้จาก host ต้นทางไปยัง host ปลายทาง

> โดยตัวทิศทางนี้สามารถเขียนได้โดยใช้ Open SSH 3.6 และไม่สามารถส่งไปยังเวอร์ชันที่เก่ากว่าได้

4.7.1.1. สร้าง SSH Key บน Host ต้นทาง

บน host ต้นทางบนผู้ใช้สำหรับที่จะใช้ในการส่งทำดังนี้ :

```
> ssh-keygen –t rsa
```

คำสั่งนี้จะมี prompt ขึ้นมาสำหรับรับ passphrase (option) และจะสร้าง 2 ไฟล์คือ id_rsa และ id_rsa.pub ซึ่งอยู่ใน ~/.ssh/.

4.7.1.2. การสำเนากุญแจสาธารณะ SSH ไปยังแต่ละ host ปลายทาง

ส่งกุญแจสาธารณะไปยังแต่ละ host ปลายทางด้วยผู้ใช้สำหรับการส่ง

```
> scp ~/.ssh/id_rsa.pub <destHost> : ~ (enter password)
```

สร้าง authorized_key ไฟล์บนแต่ละ host ปลายทาง

```
> ssh <destHost> (enter password)
> cat id_rsa.pub >> .ssh/authorized_keys
```

(ถ้าหาก แฟ้มข้อมูล .ssh ยังไม่ถูกสร้างขึ้นมา, สร้างและกำหนดให้คุณสมบัติเป็น 700 ["mkdir .ssh , chmod 700 .ssh"])

```
> chmod 600 .ssh/authorized_keys
> rm id_rsa.pub
```

4.7.1.3. ปรับแต่ง SSH Daemon บนแต่ละ host ปลายทาง

การปรับแต่งบางอย่างของ SSH Daemon นั้นอาจจะต้องการบน host ปลายทาง (เพราะว่าไม่เป็นแบบนี้เสมอ ข้ามไปยังขั้นตอนที่ 4 และทดสอบการเปลี่ยนแปลง ถ้าการทดสอบล้มเหลวให้กลับมาทำที่ขั้นตอนนี้และลองทดสอบใหม่อีกครั้ง) โดยทั่วไปแล้วขั้นตอนนี้จะสำเร็จโดย

การแก้ไขไฟล์ "/etc/ssh/sshd_config" (จำเป็นจะต้องใช้ root ในการแก้ไข) ในการตรวจสอบความถูกต้องในการปรับแต่ง ให้ดูที่คุณสมบัติดังต่อไปนี้ว่าถูกเปิดใช้หรือไม่

```
RSAAuthentication   yes
PubkeyAuthentication yes
```

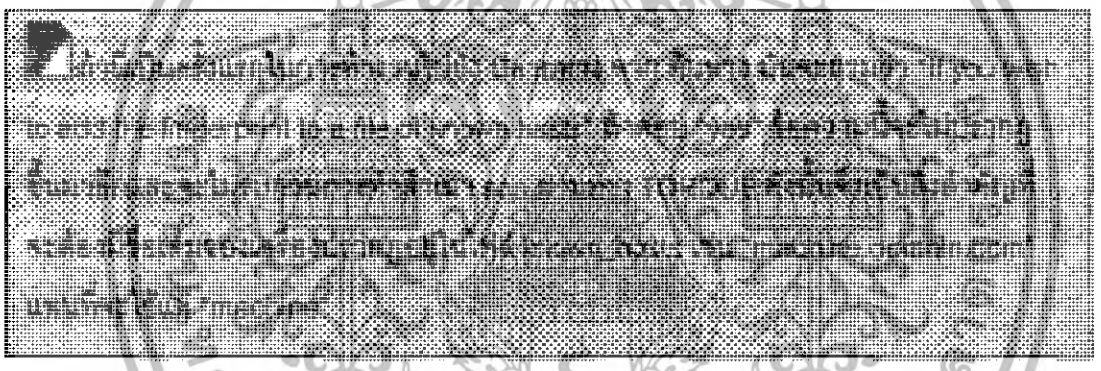
ถ้าจำเป็นต้องเปลี่ยนแปลงค่า SSH Daemon นั้นจะต้องทำการ restart (ต้องใช้ root ทำ)

```
> /etc/init.d/sshd restart
```

4.7.1.4. ตรวจสอบความถูกต้องของการปรับแต่ง SSH

ถ้าหากมีการปรับแต่งค่าทุกอย่างแล้ว คำสั่งต่อไปนี้จะถูกพิมพ์ออกมาบน host ต้นทางโดยสมบูรณ์และไม่มีการถาม password :

```
> scp <destHost>:/etc/motd /tmp
```



4.7.1.5. การเปิดใช้งาน SCP 2 ทิศทาง

จากขั้นตอนที่ได้กล่าวมาข้างต้นนั้นเป็นการอนุญาตให้ ต้นทาง กระทำกับ ปลายทาง โดยไม่ต้องใช้ password ในทางกลับกันทำขั้นตอนข้างต้นอีกครั้งแต่คราวนี้ใช้ ปลายทาง แทนที่ ต้นทาง ในการเปิดใช้งาน SCP 2 ทิศทาง (เช่น ต้นทาง สามารถส่งไปยัง ปลายทาง และ ปลายทาง สามารถส่งไปยัง ต้นทาง โดยไม่ต้องใช้ password)

4.7.1.6. คอมไพล์ TORQUE ให้สนับสนุน SCP

TORQUE ต้องทำการปรับแต่งใหม่ (จากนั้นสร้างใหม่) เพื่อใช้ SCP โดยเพิ่ม –with-scp flag เข้าไปใน configure script

```
> ./configure --prefix=xxx --with-scp
> make
```

4.7.2. NFS และ Networked Filesystem อื่นๆ

4.7.2.1. TORQUE Data Management

เมื่อชุดของงานเริ่มทำงานไฟล์ stdin จะถูกสำเนาจากการเสนอแฟ้มข้อมูลบน remote host และไฟล์นี้จะถูกวางที่แฟ้มข้อมูล "$PBSMOMHOME" บนเครื่องแม่ที่เหนือกว่า (เช่น "/usr/spool/PBS/spool") ขณะที่งานทำงาน stdout และ stderr จะถูกสร้างและถูกวางในแฟ้มข้อมูลนี้โดยใช้ชื่อ $JONID.OU และ $JOBID.ER

เมื่อ งาน ทำงานสำเร็จ MOM จะทำการ สำเนา ไฟล์ต่างๆไปยัง แฟ้มข้อมูล ที่ทำการเสนองาน โดยทั่วไปแล้วไฟล์เหล่านี้จะถูก ทำสำเนา โดยใช้ RCP หรือ SCP

ถ้ามีการเปิดใช้ filesystem เช่น NFS, DFS, หรือ AFS ซึ่งสามารถระบุให้ MOM ใช้ประโยชน์โดยระบุ "$usecp" ซึ่งอยู่ใน MOM "config" file ("$PBSMOMHOME/mom_priv") ใช้รูปแบบดังนี้ $usecp <HOST>:<SRCDIR> <DSTDIR>

mom_priv/config

```
# /home is NFS mounted on all hosts
  $usecp *:/home  /home
  # submission hosts in domain fte.com should map '/data' directory on submit host
to
  # '/usr/local/data' on compute host
  $usecp *.fte.com:/data  /usr/local/data
```

สำหรับเหตุผลใดๆก็ตาม MOM deamon ไม่สามารถที่จะ ทำสำเนา output หรือ error file ไปยัง แฟ้มข้อมูลที่เสนอมาซึ่งไฟล์นี้จะถูกสำเนาไปยัง "$PBSMOMHOME"

4.7.3. File Stage-In/Stage-Out

ความต้องการของ File stageing คือการระบุโดยใช้ stagein และ stageout ของคำสั่ง qsub โดยที่ Stagein ก่อนที่ งาน จะทำงาน ขณะที่ Stageout เกิดขึ้นหลังจาก งาน ทำงานเสร็จ

หลังจากที่ งาน ทำงานเสร็จ stagein และ stageout ทุกๆไฟล์จะถูกนำออกจากระบบ file_list อยู่ในรูปของ local_file@hostname:remote_file[,...] โดยไม่คำนึงถึงทิศทางของการ ทำสำเนา local_file คือชื่อของไฟล์บนระบบที่ถูกดำเนินการซึ่งอาจจะเป็น path เต็ม หรือ ชื่อที่สัมพันธ์กับ home directory ของผู้ใช้นั้น remote_file คือชื่อปลายทางบน host ที่ถูกกำหนดโดย hostname ซึ่งอาจจะเป็นชื่อเต็มหรือชื่อที่สัมพันธ์กับ home แฟ้มข้อมูล ของผู้ใช้นั้นบน host ปลายทาง

ชื่อของไฟล์จะถูกจับคู่กับ remote copy program (rcp/scp/cp ขึ้นอยู่กับการปรับแต่ง) ในรูปแบบต่อไปนี้:

- stagein: rcp/scp hostname:remote_file local_file
- stageout: rcp/scp local_file hostname:remote_file

ตัวอย่าง :

```
# stage /home/john/input.txt from node13 to /home/john/input.txt on master compute
node
> qsub -l nodes=1,walltime=100 -W stagein=input.txt@node13.fsc:/home/john/input.txt
```

```
# stage /tmp/output.txt on master compute node to /home/bill/output.txt on node15.fsc
> qsub -l nodes=1,walltime=100 -W
stageout=/tmp/output.txt@node15.fsc:/home/bill/output.txt
```

4.8. Interfacing with message passing

4.8.1. MPI Support (Message Passing Interface)

4.8.1.1. MPI Overview

ไลบรารี่ของการส่งข้อความนั้น ถูกใช้โดยการทำงานแบบขนานในการขยายการติดต่อระหว่างการทำงานที่ถูกกระจายทั่วคลัสเตอร์ TORQUE สามารถทำงานด้วยไลบรารี่ของการส่งข้อความใดๆและกำหนดขีดจำกัดของการรวบรวมด้วย MPI ไลบรารี่บางตัว

4.8.1.2. MPICH

หนึ่งใน MPI ไลบรารี่ที่โด่งดังที่สุดคือ MPICH หาได้จาก Argonne National Lab ซึ่งถ้าใช้ตัวนี้คุณอาจจะต้องพิจารณาเครื่องมือ mpiexec สำหรับเริ่มใช้โปรแกรม MPI ซึ่งการสนับสนุนสำหรับ mpiexec นั้นถูกรวมอยู่ใน TORQUE

MPIExec Overview

mpiexec คือ โปรแกรมที่มาแทนที่ mpirun ซึ่งเป็นส่วนหนึ่งใน MPICH มันถูกใช้ในการเตรียมการทำงานแบบขนานจาก PBS หรือการกระทำกับสิ่งแวดล้อมต่างๆ mpiexec ใช้ไลบรารี่การจัดการงานของ PBS ในการสร้างสำเนาของการประมวลผลบน node ในส่วนที่จัดสรรไว้ของ PBS

เหตุผลที่ใช้ mpiexec แทน mpirun หรือเครื่องมือจากภายนอก (mpd):

- เริ่มการทำงานด้วย TM นั้นเร็วกว่าการเรียก rsh แต่ละงาน (1 ครั้งต่อ 1 การดำเนินการ)

- ทรัพยากรต่างๆที่ถูกใช้โดยกระบวนการสร้างจะถูกประเมินค่าด้วย mpiexec และรายงานไปยัง PBS log เพราะทุกๆกระบวนการของการทำงานแบบขนานยังคงอยู่ใต้การควบคุมของ PBS ไม่เหมือนกับการใช้ mpirun-like
- งานที่นอกเหนือกำหนดขอบเขตของ cpu time, wallclock time, การใช้หน่วยความจำหรือ เนื้อที่ว่างจะถูกทำลายโดย PBS ซึ่งเป็นการยากสำหรับการดำเนินการที่ออกนอกการควบคุมของการจัดการทรัพยากรเมื่อใช้ mpiexec
- คุณสามารถใช้ mpiexec ในการบังคับใช้นโยบายความปลอดภัย ถ้าทุกๆงานถูกบังคับให้ใช้ mpiexec และสภาพแวดล้อมการดำเนินการของ PBS ซึ่งทำให้ไม่จำเป็นต้องเปิดใช้งาน rsh หรือ ssh เพื่อเข้าไปยังหน่วยประมวลผลในคลัสเตอร์

4.9. Managing Resource

4.9.1. Monitoring Resource

4.9.1.1. Resource Overview

การทำงานหลักของการจัดการทรัพยากรใดๆคือการสังเกตสถานะ, สภาพ, การปรับแต่งและการให้ประโยชน์ของการจัดการทรัพยากรซึ่ง TORQUE นั้นถูกออกแบบมาเพื่อสังเกตหน่วยประมวลผลที่อยู่ในสภาพแวดล้อมเดียวกันโดยเฉพาะ อย่างไรก็ตามทรัพยากรเหล่านี้สามารถรวมเข้ามายังคลัสเตอร์โดยใช้ระบบ scheduling ซึ่งสามารถแบ่งได้เป็น 3 หมวดหมู่คือ configuration, utilization, และ state

4.9.1.2. Configuration

Configuration นี้รวมเข้าไว้ด้วยการตรวจสอบฮาร์ดแวร์ และ การกำหนดคุณลักษณะ

คุณลักษณะ	คำอธิบาย	รายละเอียด
Architecture (arch)	ระบบปฏิบัติการของ node	ค่าที่ถูกรายงานออกมาคือค่าที่เอามาจากการติดตั้งระบบปฏิบัติการ
Memory (physmem)	RAM	RAM จะถูกรายงานออกมาในหน่วยของ กิโลไบท์
Processors (ncpus/np)	หน่วยประมวลผลจริง/เสมือน	จำนวนของหน่วยประมวลผลกลางที่ถูกตรวจได้จาก TORQUE จะถูกรายงานผ่านทางคุณลักษณะของ ncpus
Swap (totmem)	หน่วยความจำเสมือน/Swap	หน่วยความจำเสมือนจะถูกรายงานออกมาในหน่วยของกิโลไบท์

4.9.1.3. Utilization

Utilization รวมเข้าไว้ด้วยข้อมูลที่พิจารณาจากจำนวนของทรัพยากรของ node ที่มีอยู่ในขณะนั้น

คุณลักษณะ	คำอธิบาย	รายละเอียด
Disk (size)	เนื้อที่ว่างที่มีอยู่	โดยปกติแล้วพื้นที่จะไม่ถูกสังเกต ถ้าปรับแต่ง mom พารามิเตอร์ size เป็นใช้ TORQUE จะรายงานการปรับแต่งและขนาดของพื้นที่ในหน่วยกิโลไบท์
Memory (availmem)	หน่วยความจำจริง/ RAM	หน่วยความจำจริงหรือ RAM ที่มีอยู่จะถูกรายงานออกมาในหน่วยของ กิโลไบท์
Network (netload)	การใช้งาน network adapter	รายงานจำนวนของไบท์ที่ถูกส่งผ่านเข้า หรือ ออกโดย network adapter
Processor Utilization (loadave)	ภาระงานของ CPU ของ node	รายงานค่าเฉลี่ยของภาระงานในช่วงเวลา 1 นาทีที่ผ่านมา

4.9.1.4. State

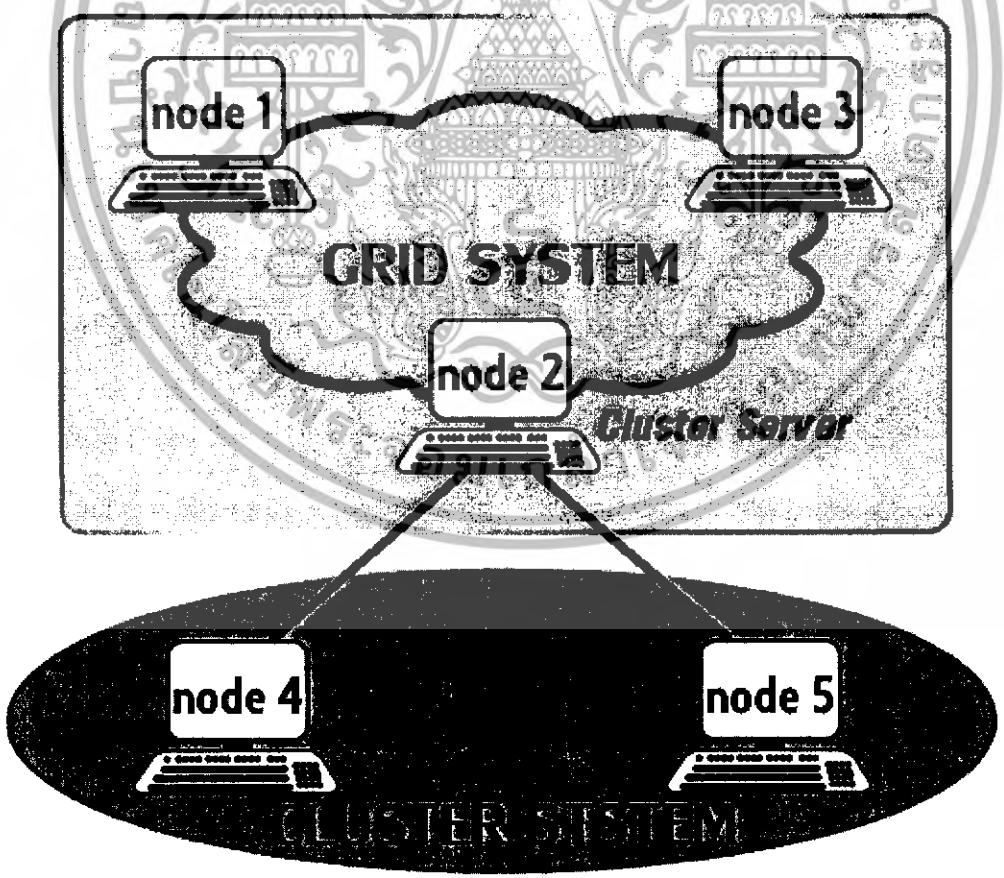
คุณลักษณะ	คำอธิบาย	รายละเอียด
Idle Time (idletime)	เวลาตั้งแต่ตรวจจับการกระทำของ keyboard/mouse ได้	เวลาในหน่วยของวินาทีตั้งแต่ตรวจจับการกระทำของ keyboard/mouse ได้
State (state)	สถานะของ node	Node สามารถเป็นได้เพียง 1 หรือมากกว่าสถานะดังนี้ • Busy – node นั้นเต็มแล้วและไม่สามารถรับงานเพิ่มได้อีก • Down – node นั้นไม่สามารถรายงานได้หรือตรวจจับได้ว่ามีการล้มเหลวของทรัพยากรหรือการปรับแต่ง node นั้นๆ • Free – node นั้นพร้อมที่จะรับงานเพิ่มเติมได้ • Offline – node นั้นถูกสั่งจากผู้ดูแลไม่ให้รับงานเพิ่มอีกต่อไป • Unknown – หา node นั้นไม่เจอ

		• Job-exclusive - หน่วยประมวลผลเสมือนทั้งหมดถูกกำหนดให้ทำงานอยู่

บทที่ 5
การทดสอบ
แอพพลิเคชั่นกระจายงานเพื่อการเข้ารหัส

5.1. ภาพรวมของระบบ

- Grid System จะมีทั้งหมด 3 เครื่องในระบบ และ 1 ในนั้นเป็น Cluster Server
- Cluster System มีทั้งหมด 2 เครื่องซึ่งติดต่อกับ Cluster Server ที่อยู่ในระบบ Grid
- การส่งงาน จะทำการส่งไฟล์ movie และ คำสั่งไปยังแต่ละเครื่องในระบบและทำการ encode โดยใช้ mencoder ซึ่งเป็น tool ที่ใช้ในการ encode
- การส่งงาน เป็นการส่งแบบ Round-Robin ซึ่งจะวนรอบส่งงานไปยังเครื่องที่ว่างโดยที่
  - เครื่องที่อยู่ในระบบ Grid จะดูว่างานเสร็จแล้วหรือไม่
  - เครื่องที่อยู่ในระบบ Cluster จะดูว่างานเสร็จแล้วหรือไม่ และ CPU Load น้อยกว่า



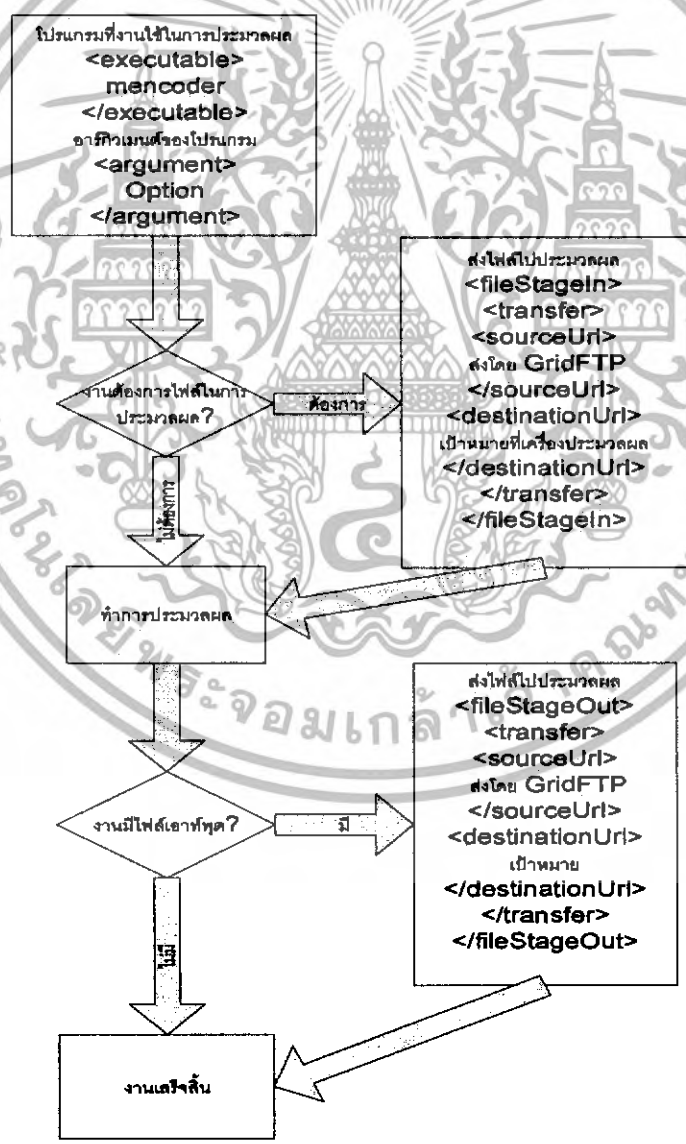
รูปที่ 5.1 ภาพรวมของระบบ

5.2. การเขียนลักษณะงานของกริดและคลัสเตอร์ (Job description)

การเขียนลักษณะงานในระบบจะแบ่งออกเป็น 2 ลักษณะ คือ การเขียนลักษณะงานในระบบกริดและการเขียนลักษณะงานในระบบคลัสเตอร์ ซึ่งการทำงานในทั้ง 2 ลักษณะนี้มีความคล้ายคลึงกันเพียงแต่การเขียนแตกต่างกันออกไปตามรูปแบบของภาษาที่ใช้เขียน ดังนี้

5.2.1. การเขียนลักษณะงานในระบบกริด

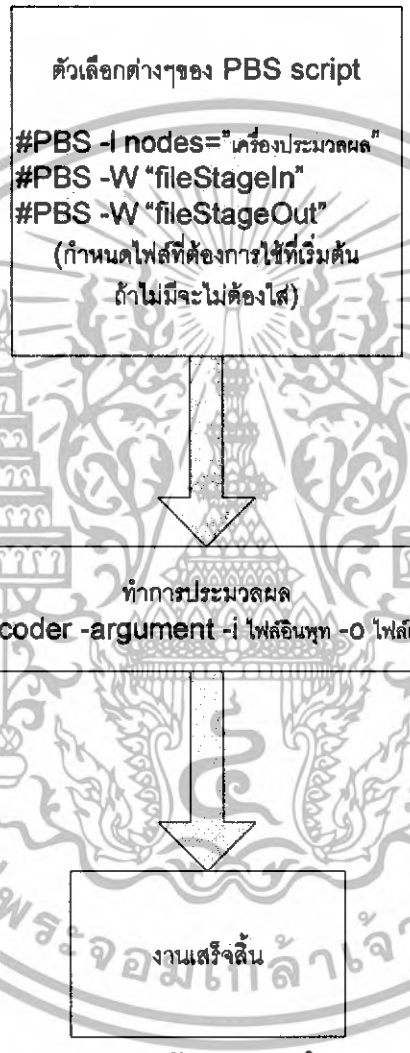
ภาษาในการเขียนลักษณะงานของระบบกริดจะใช้ภาษา RSL (Resource Specification Language) ซึ่งใน Globus toolkit รุ่น 3 ขึ้นไปจะเปลี่ยนรูปแบบของภาษาเป็น XML ทั้งหมด โดยลักษณะการเขียนนั้นได้กล่าวมาแล้วในบทที่ 3 ที่หัวข้อ GRAM ดังนั้นในหัวข้อนี้จะทำการอธิบายขั้นตอนการทำงานและการเขียนภาษาอธิบายลักษณะงานที่ใช้ในโครงงานครั้งนี้ โดยลักษณะการทำงานของลักษณะงานในระบบกริด เขียนเป็น Flow Chart ได้ ดังนี้



รูปที่ 5.2 Flow Chart ลักษณะงานในระบบกริด

5.2.2. การเขียนลักษณะงานในระบบคลัสเตอร์

โครงงานนี้เลือกใช้เครื่องมือ TORQUE ซึ่งเป็น scheduler ที่ทำงานจัดการคลัสเตอร์แบบ PBS คือ Portable Batch System ดังนั้น เครื่องมือนี้จึงมีการเขียนลักษณะงานเหมือนกับ OpenPBS,PBSpro ซึ่งจะใช้ PBS script เขียนอธิบายลักษณะงานที่จะส่งเข้าไปประมวลผลที่เครื่องต่างๆในระบบคลัสเตอร์ โดยการเขียนและการทำงานของลักษณะงานในระบบคลัสเตอร์เป็นดังนี้



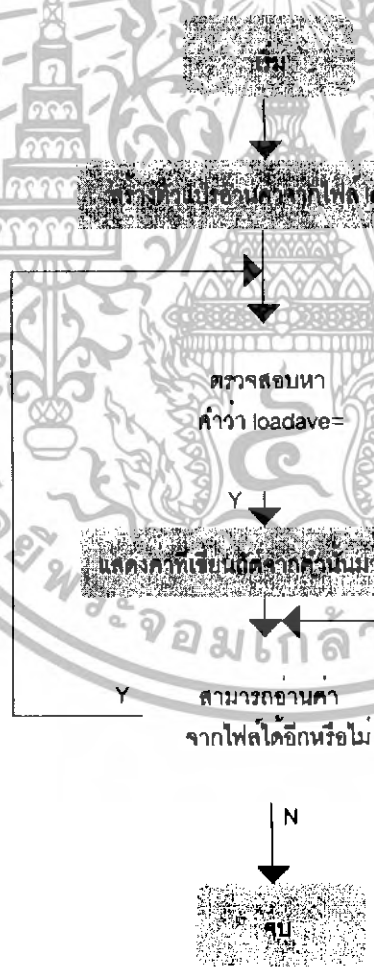
รูปที่ 5.3 Flow Chart ลักษณะงานในระบบคลัสเตอร์

เนื่องจากในแอพพลิเคชั่นที่สร้างขึ้นจำเป็นต้องมีไฟล์ที่ส่งเข้าเป็นอินพุต เพื่อนำไปทำการเข้ารหัสให้เป็น MPEG4 ดังนั้น การประมวลผลที่เครื่องอื่นจะต้องมีการส่งไฟล์จากเครื่องที่ทำการร้องขอการประมวลผล ซึ่งในที่นี้เครื่อง node 2 จะเป็นแหล่งรวมของไฟล์ทั้งหมดส่งให้เครื่องประมวลผล การเขียนลักษณะของงานของกริดจึงต้องมีการส่งไฟล์ผ่าน GridFTP โดยอิลีเมนต์หลักๆที่ใช้ คือ <filestagein> ซึ่งใช้ในการส่งไฟล์เข้าไปเพื่อประมวล และ <filestageout>

เพื่อรับไฟล์ที่ประมวลผลเสร็จแล้วกลับมาที่เครื่องที่ทำการร้องขอ (node 2) จากนั้นไฟล์ที่ส่งไปประมวลผลจะต้องลบออกจากเครื่องประมวลผลด้วย ส่วนของระบบคลัสเตอร์จะมีความคล้ายคลึงกันแต่การส่งไฟล์ไปยังเครื่องประมวลผลจะไม่ใช้ GridFTP จะใช้การคัดลอกไฟล์ซึ่งก่อนการส่งไฟล์โดยใช้ fileStageIn นั้นผู้ใช้ของแต่ละเครื่องจะต้องทำการตั้งค่าของไฟล์ .rhosts ที่อยู่ในโฟลเดอร์ Home Directory โดยใส่โฮสท์เนมและชื่อผู้ใช้ที่อนุญาตให้เข้ามาคัดลอกไฟล์ได้ก่อน ไม่เช่นนั้นจะไม่สามารถส่งไฟล์ไปประมวลผลได้

5.3 การเขียนโปรแกรมเพื่อส่งงานไปประมวลผล

ในการเขียน Shell Script เพื่อประมวลผลงานนั้นจะมีไฟล์ .sh สำคัญอยู่ 2 ไฟล์ และไฟล์ java อีก 1 ไฟล์โดยไฟล์ checkload.java จะทำการอ่านค่า status จากไฟล์ load.txt ขึ้นมา แล้วเอาค่า loadave มาแสดง ส่วนไฟล์ loadbalance.sh นั้นจะใช้ตรวจสอบว่าเครื่องไหนที่มี loadave น้อยกว่ากัน และสุดท้ายคือไฟล์ JobDistribute.sh ซึ่งเป็นสคริปต์หลักที่ใช้ในการส่งงานทั้งหมด



รูปที่ 5.4 Flow Chart **ของไฟล์** checkload.java

รูปที่ 5.5 Flow Chart **ของไฟล์** loadbalance.sh

pseudo code ของไฟล์ JobDistribute.sh

```
while [ JobID มีค่าน้อยกว่า allcom ]
do
        เพิ่มค่า JobID ทีละ 1 ;
        กำหนด JobID ให้กับ comJobX โดย X คือหมายเลขเครื่องที่ส่งงานไปรัน ;
        if [เครื่องที่รับงานเป็นเครื่องในระบบกริด ]
        then
                แก้ไขไฟล์ rsl ใหม่ ;
                ส่งงานไปรันด้วยคำสั่ง globusrun ;
        else
                แก้ไขไฟล์ shell ที่ใช้ส่งงานใหม่ ;
                ส่งงานไปรันด้วยคำสั่ง qsub ;
        fi
        เปลี่ยนหมายเลขเครื่องให้เป็นเครื่องที่ต้องการจะส่งงานเป็นลำดับถัดไป ;
done

while [ JobID มีค่าน้อยกว่า goal ]
do
        if [เครื่องที่รับงานเป็นเครื่องในระบบกริด ]
        then
                if [ถึงคิวส่งงานให้กับเครื่อง X และ งานในเครื่อง X รันเสร็จแล้ว ]
                then
                        เพิ่มค่า JobID ทีละ 1 ;
                        comIDX = JobID ;
                fi
                if [เครื่อง X เป็นเครื่องส่งงาน ]
                then
                        เพิ่มค่า JobID ทีละ 1 ;
                fi
                แก้ไขไฟล์ rsl ที่ใช้ในการส่งงานใหม่ ;
```

```
                ส่งงานไปประมวลผลยังเครื่อง X ด้วยคำสั่ง globus ;
        fi
        if [เครื่องที่รับงานเป็นเครื่องในระบบคลัสเตอร์ ]
        then
                load = loadbalance.sh ;
                if [ งานที่เครื่องหมายเลข load ประมวลผลเสร็จ ]
                then
                        comIDX = JobID ;
                        แก้ไขไฟล์ Shell Script ที่ใช้ส่งงาน
                        ส่งงานด้วยคำสั่ง qsub
                fi
        fi
        เปลี่ยนหมายเลขเครื่องให้เป็นเครื่องที่ต้องการจะส่งงานเป็นลำดับถัดไป ;
done
```

5.4. การทดสอบความสามารถของแอพพลิเคชั่น

ในการทดลองนี้ได้นำไฟล์มัลติมีเดียร์ขนาด 158.3 MB ซึ่งมีความยาวประมาณ 5 นาทีมาตัดแบ่งเป็นไฟล์ย่อยๆได้ 6 ไฟล์แล้วทำการประมวลผลแอพพลิเคชั่นกระจายงานส่งงานเข้ารหัสไปที่เครื่องต่างๆในระบบ

5.4.1. สิ่งที่ทำการทดสอบ

การทดสอบแอพพลิเคชั่นในโครงงานนี้เน้นเรื่องของเวลาในการประมวลผลของระบบกริดและคลัสเตอร์ว่าเวลาที่ใช้ในการประมวลผลเป็นอย่างไรเมื่อเปรียบเทียบกับการประมวลผลงานนี้ปกติในเครื่องเดียว

5.4.2. สภาพแวดล้อมของการทดลอง

เนื่องจากยังมีปัจจัยอื่นที่อาจมีผลทำให้ผลการทดลองไม่คงที่ จึงทำการตั้งค่าของเครื่องแต่ละเครื่องให้ปิดการทำงานของแอพพลิเคชั่นอื่นๆที่ไม่เกี่ยวข้องก่อนทำการทดลอง เพื่อความคงที่ของข้อมูลที่จะได้จากการทดลอง

5.4.3. วิธีการทดลองการทำงานของแอพพลิเคชั่น

- ทำการตัดไฟล์ชุดแรกเป็น 6 ไฟล์ย่อย ความยาวไฟล์ละ 1 นาที
- ทำการประมวลผลพร้อมจับเวลาในการประมวลผลจนกระทั่งไฟล์ที่ได้รับการเข้ารหัสเป็น MPEG4 ส่งกลับมาที่เครื่องที่ทำการร้องขอทั้งหมด

- ทดสอบแบบเดิมซ้ำ 3 ครั้ง แล้วคำนวณหาค่าเวลาเฉลี่ยของการประมวลผลที่ได้
- จากนั้นทำการเพิ่มจำนวนไฟล์เป็นเท่าตัว แล้วทดสอบแบบเดิมไปเรื่อยๆ จนกระทั่งครบทั้งสิ้น 30 ไฟล์
- เพิ่มจำนวนไฟล์เป็น 60 ไฟล์ในรอบสุดท้ายแล้วทำการทดสอบเหมือนข้อที่ผ่านมา
- จดบันทึกผลการทดลอง

5.5. ผลการทดสอบแอพพลิเคชั่น

จากการทดสอบการทำงานของแอพพลิเคชั่น ผลการทดลองได้นำมาเสนอรูปแบบของตารางและกราฟเส้น ดังนี้

file / minute	Run on Grid System without MDS & Ganglia			
	Round 1	Round 2	Round 3	Average
6 file	2.23	2.06	2.15	2.14
12 file	3.39	3.52	3.39	3.43
18 file	5.54	6.24	4.54	5.44
24 file	7.03	7.16	7.11	7.10
30 file	8.58	9.08	9.04	9.03
60 file	17.59	18.13	18.05	18.05
96 file	28.18	28.26	27.58	28.14

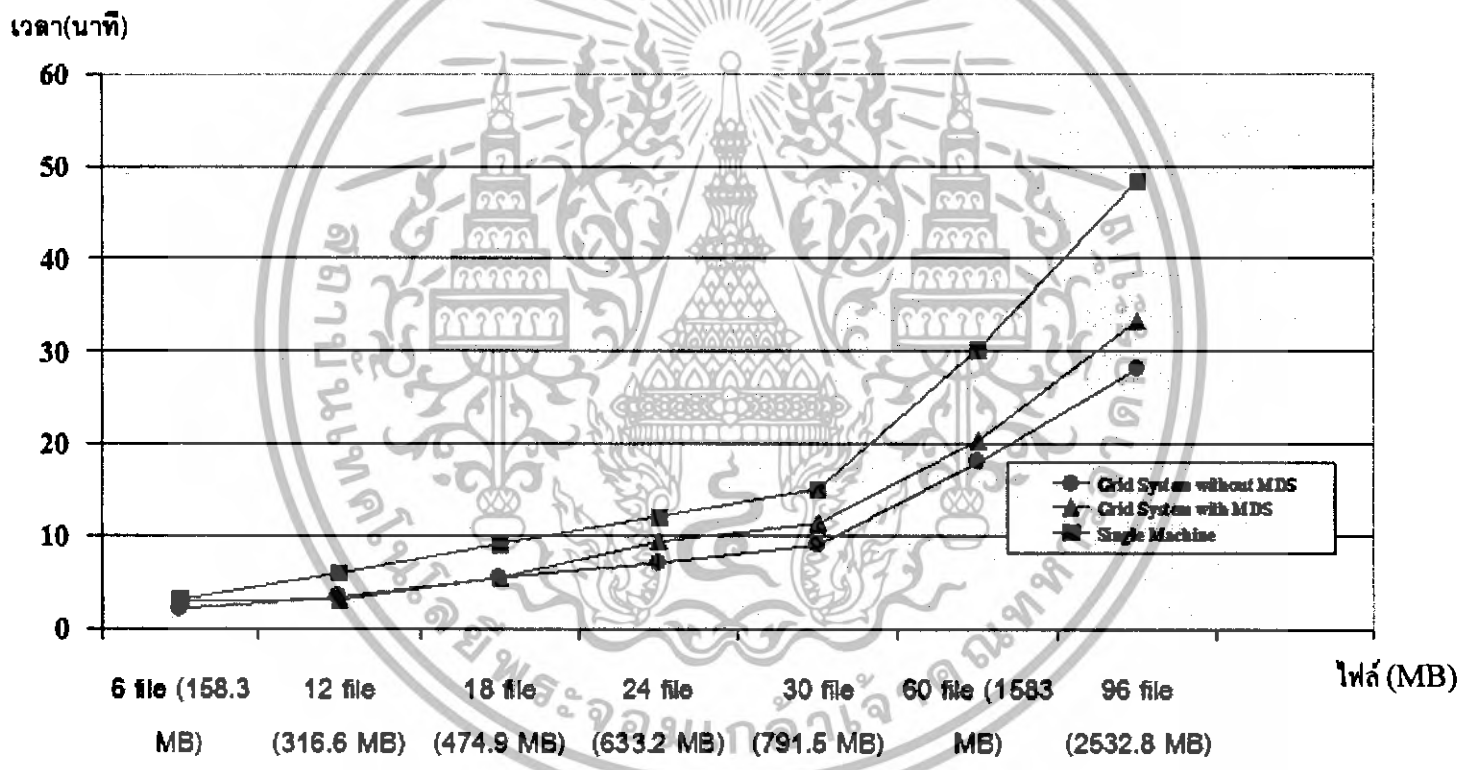
ตารางที่ 5.1 ผลการทดลองเมื่อทำงานบนระบบ Grid **โดยไม่ใช้** MDS **และ** Ganglia

file / minute	Run on Grid System with MDS & Ganglia			
	Round 1	Round 2	Round 3	Average
6 file	3.10	2.51	3.18	3.06
12 file	3.17	3.25	3.11	3.17
18 file	5.57	5.48	5.50	5.51
24 file	9.34	9.25	9.44	9.34
30 file	11.48	11.43	11.35	11.42
60 file	21.32	19.56	20.21	20.36
96 file	34.36	33.24	32.41	33.33

ตารางที่ 5.2 ผลการทดลองเมื่อทำงานบนระบบ Grid **โดยใช้** MDS **และ** Ganglia

file / minute	Run on Single Machine			
	Round 1	Round 2	Round 3	Average
6 file	3.10	3.15	3.08	3.11
12 file	6.08	6.01	5.59	6.02
18 file	9.13	9.05	9.17	9.11
24 file	12.25	12.08	12.18	12.17
30 file	15.20	15.11	15.23	15.18
60 file	30.18	30.22	30.43	30.27
96 file	48.53	48.35	48.43	48.43

ตารางที่ 5.3 ผลการทดลองเมื่อทำงานบนเครื่องเดียว



รูปที่ 5.6 กราฟแสดงผลการทดลอง

5.5.1. สรุปผลการทดลอง

การประมวลผลของแอพพลิเคชั่นที่ใช้เทคโนโลยีของกริดเข้ามาช่วย สามารถย่นระยะเวลาในการประมวลผลของแอพพลิเคชั่นปกติที่ประมวลผลบนเครื่องเดี่ยวได้ โดยที่ถ้าสังเกตจากกราฟดูจากระยะห่างของกราฟทั้งสองเส้นที่เพิ่มขึ้นเรื่อยๆแล้ว เราสรุปได้อีกข้อหนึ่งว่าจำนวนไฟล์ยิ่งมากเวลาที่ใช้จะประหยัดมากขึ้น

5.6 ประเมินผลการทดลอง

เนื่องจากมีปัจจัยอื่นๆหลายปัจจัยที่ทำให้ผลการทดลองไม่คงที่ สภาพของเครื่องแต่ละเครื่องขณะทำการประมวลผลแอพพลิเคชั่นทางกริดนั้นไม่เหมือนกัน ซึ่งเมื่อทำการทดสอบแต่ละครั้งความเร็วในการประมวลผลจึงมีระยะห่างที่ค่อนข้างมาก ดังนั้นเมื่อทำการปรับสภาพแวดล้อมของเครื่องแต่ละเครื่องในระบบ ทำให้ข้อมูลมีความคงที่มากขึ้นดังที่ได้แสดงไว้ในตารางบันทึกผล ดังนั้น ถ้างานในโครงงานฉบับนี้ได้นำไปใช้จริง ผลที่ได้จากการใช้ที่สภาพแวดล้อมต่างกัน อาจไม่เป็นไปตามผลการทดลองที่ได้ทำไว้ข้างต้น จึงยังต้องเป็นกรณีศึกษาที่ต้องนำไปพัฒนาเพิ่มต่อไป

บทที่ 6
ปัญหาอุปสรรคและแนวทางในการพัฒนาต่อ

ปัญหาและอุปสรรคส่วนใหญ่ที่เกิดขึ้นในโครงงานนี้เกิดจากการใช้เครื่องมือต่างๆ ที่มีหน้าที่การทำงานแตกต่างกันแต่ต้องนำมาใช้งานร่วมกันให้ได้ และเนื่องด้วยเครื่องมือที่นำมาใช้งานเหล่านี้มีความซับซ้อน และอยู่ในช่วงของการพัฒนาทำให้เสถียรภาพของการทำงานยังไม่ดีเท่าที่ควร โดยปัญหาหลักๆที่เกิดขึ้นแยกได้ดังหัวข้อต่อไปนี้

6.1. ปัญหาจากการใช้ระบบปฏิบัติการ Linux Fedora Core 4 บน VMware

ปัญหาที่เกิดจากการใช้ระบบปฏิบัติการ

- ปัญหาจากความไม่เสถียรของระบบปฏิบัติการ โดยที่บางครั้งขณะรันโปรแกรมอยู่ ตัว explorer ของลินุกซ์จะค้าง ทำให้ไอคอนต่างบน desktop หายไป และไม่สามารถใช้งาน Folder ต่างๆได้ นอกจากจะรีสตาร์ตเครื่องใหม่
- ความไม่เสถียรของระบบปฏิบัติการอีกอย่างหนึ่งก็คือ บางครั้งหากเราปิดเครื่องด้วยคำสั่ง Suspend พอจะมากด Resume เพื่อเข้ามาใช้งาน ปรากฏว่าระบบปฏิบัติการนั้นไม่สามารถเปิดได้
- ปัญหาจากการ shutdown เครื่อง โดยหากทำการ shutdown เครื่อง ค่า Environment ต่างๆ ที่เคยตั้งค่าไว้จะถูกลบหายไป ต้องตั้งค่าใหม่ทุกครั้งที่เปิดเครื่อง

แนวทางการแก้ไขปัญหา

- หาก Suspend แล้วไม่สามารถเปิดเครื่องได้ ก็ควรที่จะทำการ backup ระบบไว้ตลอดเวลา หากระบบไม่สามารถใช้การได้ จะได้นำตัว backup มาใช้ได้ทันที
- หากจำเป็นที่จะต้อง shutdown เครื่อง ให้ทำการบันทึกคำสั่งที่ไว้ใช้ในการตั้งค่า Environment ไว้ในไฟล์ไดไฟล์หนึ่งก่อน เวลาเปิดเครื่องมาใหม่จะได้ไม่ต้องมานั่งพิมพ์คำสั่งเองใหม่ทั้งหมด

6.2. ปัญหาจากการใช้ Globus Toolkit 4.0.1

6.2.1. GRAM (Grid Resource Allocation Manager)

ปัญหาเชิงเทคนิคที่พบเมื่อใช้ GRAM

- ปัญหาจากความไม่เสถียรของการส่งงานด้วยคำสั่ง globus-run เมื่อมีการใช้ GridFTP เข้าร่วมด้วยโดยปัญหานี้แยกออกเป็นหลายประเภท เช่น บางครั้งอาจเกิด filestage in error หรือ Unsubmitted Job เมื่อใช้งานร่วมกับ PBS

- ปัญหาความยุ่งยากในการตั้งค่าของระบบก่อนการใช้งาน และขณะใช้งานเนื่องจากขั้นตอนมีความซับซ้อนและละเอียดอ่อน
- ขณะทำงานอาจเกิด container fatal error ซึ่งเป็นปัญหาหนักที่ทำให้ไม่สามารถทำการ start container และไม่สามารถเรียกใช้ service ของ GRAM ได้

แนวทางการแก้ไข

- ปัญหา filestagein error จะมีสาเหตุหลายประการ อันดับแรกควรตรวจสอบทุกครั้งว่าผู้ใช้ทำการรันคำสั่ง grid-proxy-init เรียบร้อยแล้วจากนั้นถ้าปัญหายังคงอยู่ให้ทำการหยุดการทำงานของคอนเทนเนอร์ แล้วตรวจสอบการตั้งค่าที่ขั้นตอนของ postgres database ซึ่งกล่าวไว้ที่ภาคผนวกให้เรียบร้อย จากนั้นทำการเริ่มการทำงานของ gridftp server แล้วจึงทำการเริ่มการทำงานของคอนเทนเนอร์อีกครั้งหนึ่ง โดยจะมีขั้นตอนการ test filestagein ให้ทำขั้นตอนนั้นให้ทำงานได้ก่อนจึงจะสามารถใช้งานได้ ในส่วนของปัญหา Unsubmitted job นั้นมีความซับซ้อนในการแก้ปัญหา โดยส่วนนี้เกิดจากเป็น bug ของ globus toolkit ในเรื่องของการรับค่าเวลาของส่วน Globus toolkit และส่วนของ PBS นั้นไม่ตรงกันซึ่งเมื่อทำตามขั้นตอนของการแก้ก็อาจจะทำงานไม่ได้ ดังนั้น ก่อนเกิดปัญหาควรทำการสำรองไฟล์ VMware ไว้ก่อน
- เนื่องจากระบบการติดตั้งเครื่องมือนั้นมีความละเอียดซับซ้อน ดังนั้น การตั้งค่าต้องทำให้ทำงานได้ทุกขั้นตอนก่อนจึงจะสามารถใช้งานได้ ควรละเอียดรอบคอบในการตั้งค่า
- ปัญหาของ fatal error นั้นยังไม่มีทางแก้ไขนอกจากสำรองไฟล์ไว้ก่อน เนื่องจากปัญหานี้ทำให้ระบบทุกอย่างใช้ไม่ได้ทั้งหมด ถ้าเกิดขึ้นอาจต้องลงใหม่ตั้งแต่ลงระบบปฏิบัติการ Linux

6.2.2. MDS (Monitoring and Discovery Service)

ปัญหาเชิงเทคนิคที่พบเมื่อใช้ MDS

- จำนวนไฟล์ configuration มีจำนวนมากทำให้ยากต่อการปรับแต่งรายละเอียดต่างๆ
- ใช้เวลาในการ query ข้อมูลนาน ส่งผลให้การทำงานในระบบรวมช้าลงไปด้วย
- ไฟล์ configuration ทั้งหมดเป็นเอกสาร XML ซึ่งมี Attribute หลากหลาย
- หากไม่ทำการ sync clock (ทำให้เวลาเดินตามจริง) จะส่งผลให้ทำการ query ไม่ได้
- ต้องใช้ user ที่ได้ทำ Certificate ในการ query ข้อมูลเท่านั้น
- ในเวอร์ชันที่ใช้นั้น (globus toolkit 4.0.1) เป็นเวอร์ชันที่ออกใหม่ทำให้ค้นหาข้อมูล, ตัวอย่างได้ค่อนข้างยาก และ ยังมีผู้ที่ชำนาญทางด้านนี้น้อย

แนวทางการแก้ไขปัญหา MDS (Monitoring and Discovering System)

- ต้องมีความละเอียดรอบคอบในการศึกษาและเปลี่ยนแปลงไฟล์ configuration

- ควรศึกษา XML เบื้องต้นเพื่อให้ง่ายต่อการทำความเข้าใจ
- ทำการ sync clock ทุกครั้งก่อนที่จะทำการส่งงาน
- หากต้องการข้อมูล และ ตัวอย่างที่เพียงพอควรจะลด เวอร์ชัน ของ globus toolkit ลง

6.2.3. GridFTP

ส่วนของ GridFTP นี้จะเป็นส่วนที่ใช้ร่วมกับ GRAM ซึ่งเกี่ยวเนื่องกับ filestagein โดยตรง ดังนั้น ส่วนนี้จึงเป็นส่วนสำคัญที่ต้องตั้งค่าให้สามารถทำงานได้ก่อนการใช้ในการส่งไฟล์ไปประมวลผลจริง ปัญหาที่เกิดขึ้นส่วนใหญ่ในการใช้ส่วนนี้ คือ อาจเกิดความสับสนในเรื่องของ path ที่อยู่ของทั้งแหล่งไฟล์และเป้าหมายที่ต้องเขียนลงใน RSL ซึ่งจะสับสนและถ้าใส่ path ผิดจะทำให้ไม่มีการส่งไฟล์เข้าประมวล และเกิดความสับสนได้เนื่องจาก ไม่สามารถ debug ได้ว่าผิดส่วนใด

6.3. ปัญหาจากการใช้เครื่องมือเพิ่มเติมในโครงงาน

6.3.1. Ganglia

ปัญหาเชิงเทคนิคที่พบเมื่อใช้ Ganglia (Information Provider)

- เป็นเครื่องมือที่ใช้ในการทำงานกับคลัสเตอร์
- มีปัญหาในการติดตั้งเนื่องจากมีการอ้างอิงเครื่องมือและ library อื่นๆ
- มีข้อกำหนดในการใช้ร่วมกันระหว่าง MDS และ Ganglia ซึ่งก็คือ เวอร์ชัน ของ tool จะต้องรองรับกันได้

แนวทางการแก้ไขปัญหา Ganglia

- เครื่องที่จะติดตั้ง Ganglia ทุกเครื่องควรจะอยู่ในคลัสเตอร์
- เตรียมเครื่องมือและ library อื่นๆให้พร้อมก่อนการติดตั้ง
- ศึกษาว่า Ganglia เวอร์ชันใดที่สามารถใช้ร่วมกับ globus toolkit ที่นำมาใช้ในโครงงานได้

6.3.2. TORQUE-2.0.0p7

ปัญหาเชิงเทคนิคที่พบเมื่อใช้ TORQUE

- การเขียนสคริปท์ภาษาของลักษณะงานใน PBS จะพบปัญหาเมื่อใช้ filestagein ในการส่งไฟล์มัลติมีเดียร์ไปที่เครื่องอื่นเพื่อทำการประมวลผล เนื่องจากติดปัญหาทางด้านความปลอดภัยในการส่งงาน ทำให้ไม่สามารถส่งไฟล์ได้เหมือนการใช้ filestagein ผ่าน RSL ในระบบกริดที่มีการทำ certificate และ grid-proxy-init แล้ว
- Path ในการใช้ filestagein นั้นมีความซับซ้อนและสับสนเช่นเดียวกับการใช้ผ่าน RSL

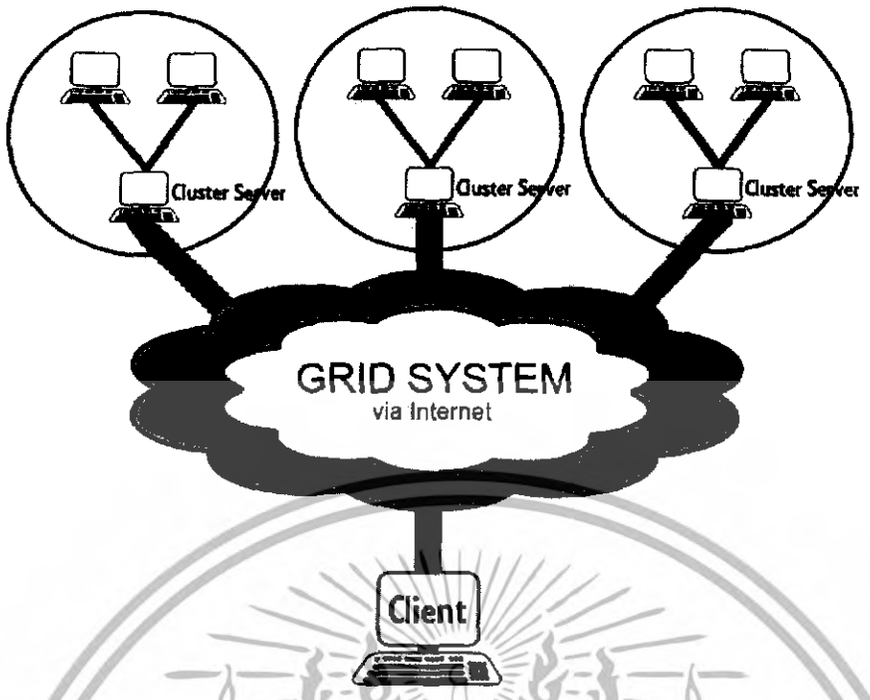
แนวทางแก้ไขปัญหา

- การแก้ปัญหาในการส่งไฟล์ของแต่ละผู้ใช้จะต้องสร้างไฟล์ .rhosts ไว้ที่แฟ้มเอกสาร Home Directory โดยภายในต้องใส่ชื่อโฮสท์เนมของแต่ละเครื่องในระบบคลัสเตอร์และผู้ใช้ของแต่ละโฮสท์เป็นการอนุญาตให้สามารถส่งไฟล์ระหว่างกันได้
- เรื่องของ Path ของแหล่งไฟล์และเป้าหมายต้องศึกษาการใช้ PBS script จากวิธีการใช้ที่กล่าวไว้ในภาคผนวกและจากเว็บไซต์ www.clusterresources.com ก่อนการใช้งานและต้องมีความรอบคอบในการใส่ path

6.4. แนวทางในการพัฒนาต่อ

โครงงานส่วนที่ทำในปีการศึกษา 2548 นั้นจะเป็นเรื่องของการทำ service ใส่เข้าไปในคอนเทนเนอร์ ซึ่งเป็น matrix service แล้วทำการเขียนแอพพลิเคชั่นกระจายงานโดยใช้จาวา เมื่อทำการประมวลผลจะมีการเรียกใช้ matrix service ที่อยู่ในเครื่องแต่ละเครื่องในระบบกริด ประมวลผลแต่ละค่าของ matrix ส่งกลับมาที่เครื่องที่เรียกใช้งาน ในส่วนของการใช้เครื่องมือที่ Globus toolkit มีให้นั้นยังไม่ได้ใช้งานในส่วนของ GRAM MDS และ GridFTP

โครงงานที่ทำในปีการศึกษานี้ (ปีการศึกษา 2549) จึงมีการนำแต่ละองค์ประกอบของ Globus toolkit มาใช้งาน ทั้ง 3 ส่วน คือ GRAM MDS และ GridFTP จากการศึกษาและทดสอบการทำงานนั้นในส่วนของ MDS ที่ใช้ในโครงงานนี้เป็นการใช้งานโดยนำ MDS และ Ganglia มาใส่ที่เครื่องในระบบกริดเครื่องเดียว คือ เครื่อง node 1 และ node 3 (จากรูปอ้างอิงในบทที่ 5 รูปที่ 5.1 ภาพรวมของระบบ) ซึ่งในความเป็นจริงแล้ว MDS และ Ganglia นั้นเป็นเครื่องมือที่ใช้งานในระบบคลัสเตอร์ เครื่อง node 1 และ node 3 นั้นสมควรที่จะต่อระบบออกเป็นอีก 2 คลัสเตอร์ ซึ่งอาจจำลองได้เป็นแต่ละหน่วยงานที่เชื่อมต่อกันด้วยระบบกริด ดังรูป



รูปที่ 6.1 ภาพรวมของระบบที่สามารถนำไปพัฒนาต่อ

บรรณานุกรม

เอกสารอ้างอิงที่เป็น Web-site

[1] Globus toolkit,

www.globus.org

[2] Globus Consortium

www.globusconsortiums.org

[3] IBM – Monitor and discover grid service

http://www-128.ibm.com/developerworks/grid/library/gr-gt4mds/?ca=dnt-637

[4] Cluster resources

http://www.clusterresources.com/wiki/doku.php?id=torque:torque_wiki

[5] Codecs and container

http://en.wikipedia.org/wiki/Codecs

http://en.wikipedia.org/wiki/Container_format_%28digital%29

[6] Digital video formats

http://www.shallowsky.com/linux/videoformats.html

[7] Shell script

www.exzilla.net/docs/unix/unixshell.pdf

[8] Ganglia installation

http://research.gc.cuny.edu/index.php/Ganglia_installation_and_configuration

[9] MDS WS GRAM configuration

https://viewcvs.globus.org/toolkit/docs/4.0/admin/docbook/ch11.html#s-wsgram-Interface_Config_Frag-autoregistration

[10] Ganglia toolkit

http://www.msg.ucsf.edu/local/ganglia/ganglia_docs/index.html

[11] Globus toolkit

https://www-unix.globus.org/toolkit/docs/development/4.1.0/info/index/developer/index.html

[12] XML

http://internet.se-ed.com/content/IN80/IN80_51.asp

เอกสารอ้างอิงที่เป็นปริญญานิพนธ์

[13] ณีรนุช สิปิพัฒนากุล, ทศพร คชานุบาล, สุรีพร องคนิกูล "การประมวลผลแบบกริด" ปริญญานิพนธ์ วิศวกรรมศาสตร์บัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ ปีการศึกษา 2548 คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคผนวก ก.

ขั้นตอนการลงระบบปฏิบัติการ Linux Fedora Core 4

ขั้นตอนการลง Linux Fedora Core 4

ในโครงงานนี้จะทำการลง Linux ใน VMware หรือ Virtual Machine ซึ่งจำลองระบบปฏิบัติการขึ้นบนวินโดว์ โดยไฟล์ ISO ที่ต้องการใช้มี ดังนี้

- FC4-i386-disc1.iso 06-Jun-2005 22:54 635M
- FC4-i386-disc2.iso 06-Jun-2005 22:55 638M
- FC4-i386-disc3.iso 06-Jun-2005 22:56 638M
- FC4-i386-disc4.iso 06-Jun-2005 22:57 630M

เมื่อใส่ไฟล์ ISO ไฟล์แรกลงไปที่มีเดียร์ของ VMware แล้วทำการเปิดเข้าสู่ระบบจะเข้าสู่การลงระบบปฏิบัติการ Linux โดยจะต้องตั้งค่าดังต่อไปนี้

Language: English (default)

Keyboard: US English (default)

Installation Type: Server

Disk Partitioning: Remove all partitions

Boot loader: GRUB (default): no boot loader password

Automatically partition under the default partition parameters

หลังจากนั้นทำการตั้งค่า IP และ DNS ดังนี้

Test1

IP: 161.246.6.112

Netmask: 255.255.255.0

Hostname: test1.kmitl.ac.th

Gateway: 161.246.6.254

Primary DNS: 161.246.4.3

Secondary DNS: 161.246.52.21

Test2

IP: 161.246.6.113

Netmask: 255.255.255.0

Hostname: test2.kmitl.ac.th

Gateway: 161.246.6.254

Primary DNS: 161.246.4.3

Secondary DNS: 161.246.52.21

Test3

IP: 161.246.6.114

Netmask: 255.255.255.0

Hostname: test3.kmitl.ac.th

Gateway: 161.246.6.254

Primary DNS: 161.246.4.3

Secondary DNS: 161.246.52.21

Firewall: No firewall - Disable SELinux

ตั้งเวลา Time Zone เป็น Asia/Bangkok

Root password: gridcomputing (ตั้งได้ตามถนัด)

Packages to select:

Xwindows

Gnome

Applications:

Editors

Graphical Internet

Text based internet

Server:

Server configuration tools

Web server

Windows File Server

Postgres SQL Database

Development:

Development tools

Java Development

System:

Admin tools

System tools

Printing Support

หลังจากตั้งค่าเสร็จแล้วระบบจะเริ่มทำการติดตั้งระบบปฏิบัติการให้ทำการเปลี่ยน ISO ไฟล์ไปเรื่อยๆจนการติดตั้งค่าเสร็จสิ้น

จากนั้นทำการสร้าง account ของผู้ใช้โดยในโครงงานนี้จะสร้าง 3 account ดังนี้

Account: nodeA ซึ่งเป็นผู้ใช้ที่อยู่ที่เครื่อง test1

Account: nodeB ซึ่งเป็นผู้ใช้ที่อยู่ที่เครื่อง test2

Account: nodeC ซึ่งเป็นผู้ใช้ที่อยู่ที่เครื่อง test3

แนะนำให้สร้างทั้ง 3 account บนเครื่องทั้ง 3 เครื่อง คือ test1,test2,test3 เพราะต้องใช้ในส่วนของ GridFTP ซึ่งเป็นทั้งที่อยู่ของไฟล์ที่ถูกส่งไปประมวลผลและเป็นที่อยู่ของผลลัพธ์ที่ออกมาเพื่อทำการส่งไฟล์กลับมาที่เครื่องที่เรียกใช้แอพพลิเคชั่น ขั้นตอนที่ต้องทำถัดไปเป็น ดังนี้

- ทำการแก้ไขไฟล์ /etc/group โดยเติมส่วนต่อไปนี้

globus:x:501:

- ใช้คำสั่ง useradd เพื่อเพิ่ม account 'globus'

[root@test1]# /usr/sbin/useradd -c "Globus User" -g 501 -m -u 501 globus

- ใช้คำสั่ง useradd เพื่อเพิ่ม account ของผู้ใช้แต่ละ node ดังไป (ทำจาก)

[root@testX]# /usr/sbin/useradd -c "nodeX User" -g 100 -m -u 101 nodeX

ตั้ง password โดยใช้คำสั่ง

passwd globus

passwd nodeX

ภาคผนวก ข.

ขั้นตอนการลงระบบกริดแบบ 3 เครื่องที่ใช้ในโครงงาน และการทำ Simple CA

ขั้นตอนการลงระบบกริดแบบ 3 เครื่อง

สิ่งที่ต้องทำหลังการติดตั้งระบบปฏิบัติการ Linux **สิ่งแรก คือ การติดตั้งและตั้งค่า** PostgreSQL Relational Database **เพื่อใช้ใน** GridFTP **และการเริ่มการทำงานของคอนเทนเนอร์ของระบบกริด โดยการตั้งค่านี้ต้องทำทั้ง** 3 **เครื่องในระบบ**
รันคำสั่งตามขั้นตอน ดังต่อไปนี้

- [root@testX ~]# rpm -qa|grep postgres

ผลลัพธ์ที่ได้

postgresql-8.0.3-1

postgresql-server-8.0.3-1

postgresql-libs-8.0.3-1

- [root@testX ~]# grep postgres /etc/passwd

ผลลัพธ์ที่ได้

postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash

- [root@testX ~]# /sbin/chkconfig --list | grep postgres

ผลลัพธ์ที่ได้

postgresql 0:off 1:off 2:off 3:off 4:off 5:off 6:off

- [root@testX ~]# mkdir -p /opt/pgsql/data

- [root@testX ~]# chown -R postgres /opt/pgsql

- [root@testX ~]# chgrp -R postgres /opt/pgsql

จากนั้นเปลี่ยนเป็น user 'postgres' ด้วยคำสั่ง

- [root@testX ~]# su – postgres

ทำการเริ่มต้นระบบฐานข้อมูล

- -bash-3.00$ /usr/bin/initdb -D /opt/pgsql/data

แล้วทำการเริ่มการทำงานของฐานข้อมูล

- /usr/bin/postmaster -i -D /opt/pgsql/data > /opt/pgsql/logfile 2>&1 & (**คำสั่งนี้ต้อง**กลับมาทำทุกครั้งถ้าปิดเครื่องแล้วเปิดใหม่)

- ps auwwwwx|grep post (**รันเพื่อตรวจสอบ** process)

หลังจากตั้งค่าฐานข้อมูลเสร็จแล้วต่อไปต้องทำการถอนการติดตั้ง Java **อันเก่าที่ติดตั้งมาพร้อมกับตัวระบบปฏิบัติการ โดยในโครงงานได้ทำการดาวน์โหลด** jdk-1_5_0_08-linux-i586.bin **มาใช้งานโดยเข้าไปที่** website http://java.sun.com/j2se/1.5.0/download.jsp **โดยคำสั่งที่ต้องทำมีขั้นตอน ดังนี้**

- [root@testX ~]# which java

ผลลัพธ์: /usr/bin/java

- [root@testX ~]# ls -alh /usr/bin/java

ผลลัพธ์: lrwxrwxrwx 1 root root 22 Oct 7 16:06 /usr/bin/java ->
/etc/alternatives/java

- [root@testX ~]# ls -alh /etc/alternatives/java

ผลลัพธ์: lrwxrwxrwx 1 root root 35 Oct 7 16:06 /etc/alternatives/java ->
/usr/lib/jvm/jre-1.4.2-gcj/bin/java

- [root@testX ~]# rpm -qf /usr/lib/jvm/jre-1.4.2-gcj/bin/java

ผลลัพธ์: java-1.4.2-gcj-compat-1.4.2.0-40jpp_31rh.FC4.2

- [root@testX ~]# yum remove java-1.4.2-gcj-compat-1.4.2.0-40jpp_31rh

- [root@testX ~]# ls -l jdk-1_5_0_08-linux-i586.bin

ผลลัพธ์: -rw-r--r-- 1 root root 48974825 Feb 20 11:13 jdk-1_5_0_08-linux-i586.bin

- [root@testX ~]# md5sum jdk-1_5_0_08-linux-i586.bin

ผลลัพธ์: 3cdad4a383b93680f02f6f06198c2227 jdk-1_5_0_08-linux-i586.bin

- [root@testX ~]# mv jdk-1_5_0_08-linux-i586.bin /opt/

- [root@testX ~]# cd /opt/

- [root@testX opt]# chmod 755 jdk-1_5_0_08-linux-i586.bin

- [root@testX opt]# ./jdk-1_5_0_08-linux-i586.bin

ตรวจสอบหลังการลงจาวาใหม่

- [root@testX opt]# ls -l /opt/jdk1.5.0_08/

- [root@testX opt]# /opt/jdk1.5.0_08/bin/java –version

- [root@testX opt]# /opt/jdk1.5.0_08/bin/javac –version

ทำการตั้งค่า path ที่จะเรียกใช้งาน

- [root@testX ~]# export JAVA_HOME=/opt/jdk1.5.0_08

- [root@testX ~]# export PATH=$JAVA_HOME/bin:$PATH

- [root@testX ~]# which java

/opt/jdk1.5.0_08/bin/java

- [root@testX ~]# which javac

/opt/jdk1.5.0_08/bin/javac

ขั้นตอนถัดไปต้องทำการติดตั้ง Ant โดยรันคำสั่งตามขั้นตอน ดังนี้

- [root@testX ~]# wget http://apache.mirrormax.net/ant/binaries/apache-ant-1.6.5-bin.tar.bz2
- [root@testX opt]# ls -l apache-ant-1.6.5-bin.tar.bz2

  -rw-r--r-- 1 root root 6743024 Jun 2 2005 apache-ant-1.6.5-bin.tar.bz2
- [root@testX opt]# md5sum apache-ant-1.6.5-bin.tar.bz2

  26031ee1a2fd248ad0cc2e7f17c44c39 apache-ant-1.6.5-bin.tar.bz2
- [root@testX ~]# mv apache-ant-1.6.5-bin.tar.bz2 /opt/
- [root@testX ~]# cd /opt/
- [root@testX opt]# tar -jxf apache-ant-1.6.5-bin.tar.bz2

หลังจากทำคำสั่งเสร็จแล้ว ทำการตั้งค่า path และตรวจสอบการใช้งาน

- [root@testX opt]# export ANT_HOME=/opt/apache-ant-1.6.5
- [root@testX opt]# export PATH=$ANT_HOME/bin:$PATH
- [root@testX opt]# which ant

  /opt/apache-ant-1.6.5/bin/ant
- [root@testX opt]# ant –version

ขั้นตอนถัดไปจะเป็นการติดตั้ง globus toolkit และการทำ CA ในส่วนของความปลอดภัย โดยขั้นตอนนี้จะตั้ง test2 เป็น CA เซอฟเวอร์ให้เครื่องอื่นทำการติดต่อเข้ามาทำการ sign certificate ให้ ส่วนของการติดตั้ง globus toolkit นั้นต้องทำทุกเครื่องในระบบ โดยมีขั้นตอนดังต่อไปนี้

- [root@testX ~]# mkdir -p /opt/globus-4.0.1
- [root@testX ~]# chown globus.globus /opt/globus-4.0.1
- [root@testX ~]# ls -alh /opt | grep globus

  drwxr-xr-x 2 globus globus 4.0K Feb 20 12:05 globus-4.0.1
- [root@testX opt]# su - globus
- [globus@testX ~]$ cd
- [globus@testX ~]$ wget http://www.globus.org/ftppub/gt4/4.0/4.0.1/installers/src/gt4.0.1-all-source-installer.tar.bz2
- [globus@testX ~]$ ls -l gt4.0.1-all-source-installer.tar.bz2

  -rw-rw-r-- 1 globus globus 88609887 Aug 5 2005 gt4.0.1-all-source-installer.tar.bz2

- [globus@testX ~]$ md5sum gt4.0.1-all-source-installer.tar.bz2

  398076812364f53d694194320836b8ad gt4.0.1-all-source-installer.tar.bz2

- [globus@testX ~]$ tar -jxf gt4.0.1-all-source-installer.tar.bz2

- [globus@testX ~]$ export JAVA_HOME=/opt/jdk1.5.0_08

- [globus@testX ~]$ export PATH=$JAVA_HOME/bin:$PATH

- [globus@testX ~]$ which java

  /opt/jdk1.5.0_06/bin/java

- [globus@testX ~]$ which javac

  /opt/jdk1.5.0_06/bin/javac

- [globus@testX ~]$ export ANT_HOME=/opt/apache-ant-1.6.5

- [globus@testX ~]$ export PATH=$ANT_HOME/bin:$PATH

- [globus@testX ~]$ which ant

  /opt/apache-ant-1.6.5/bin/ant

- [globus@testX ~]$ cd gt4.0.1-all-source-installer

- [globus@testX gt4.0.1-all-source-installer]$ export GLOBUS_LOCATION=/opt/globus-4.0.1

- [globus@testX gt4.0.1-all-source-installer]$ ./configure --prefix=$GLOBUS_LOCATION --disable-rls

  checking build system type... i686-pc-linux-gnu

  checking for javac... /opt/jdk1.5.0_08/bin/javac

  checking for ant... /opt/apache-ant-1.6.5/bin/ant

  configure: creating ./config.status

  config.status: creating Makefile

- [globus@testX gt4.0.1-all-source-installer]$ make

- [globus@testX gt4.0.1-all-source-installer]$ make install

- [globus@testX ~]$ source /opt/globus-4.0.1/etc/globus-user-env.sh

- wget http://www-unix.globus.org/ftppub/gt4/4.0/4.0.1/updates/src/globus_simple_ca-0.15.tar.gz

- wget http://www-unix.globus.org/ftppub/gt4/4.0/4.0.1/updates/src/globus_simple_ca_setup-0.27.tar.gz

- [globus@testX ~]$ gpt-build -update ./globus_simple_ca-0.15.tar.gz

- [globus@testX ~]$ gpt-build -update ./globus_simple_ca_setup-0.27.tar.gz

ขั้นตอนในการทำ simpleCA

- export GLOBUS_LOCATION=/opt/globus-4.0.1
- source $GLOBUS_LOCATION/etc/globus-user-env.sh
- [globus@test2 gt4.0.1-all-source-installer]$ $GLOBUS_LOCATION/setup/globus/setup-simple-ca

WARNING: GPT_LOCATION not set, assuming:

GPT_LOCATION=/opt/globus-4.0.1

C e r t i f i c a t e A u t h o r i t y S e t u p

This script will setup a Certificate Authority for signing Globus users certificates. It will also generate a simple CA package that can be distributed to the users of the CA.

The CA information about the certificates it distributes will be kept in:

/home/globus/.globus/simpleCA/

The unique subject name for this CA is:

cn=Globus Simple CA, ou=simpleCA-nodeb.ps.univa.com, ou=GlobusTest, o=Grid

Do you want to keep this as the CA subject (y/n) [y]:

n

Enter a unique subject name for this CA:

cn=<MyOrganization>,ou=ConsortiumTutorial,ou=GlobusTest,o=Grid

Replace <MyOrganization> with the name of your organization.

Enter the email of the CA (this is the email where certificate requests will be sent to be signed by the CA):

<MyEmailAddress>

Replace <MyEmailAddress> with your e-mail address, or simply a dummy e-mail address as this is not important.

The CA certificate has an expiration date. Keep in mind that once the CA certificate has expired, all the certificates signed by that CA become invalid. A CA should regenerate the CA certificate and start re-issuing ca-setup packages before the actual CA certificate expires. This can be done by re-running this setup script. Enter the number of DAYS the CA certificate should last before it expires.

[default: 5 years (1825 days)]:

Hit Return to accept default.

Enter PEM pass phrase:<MyPEMpassPhrase>

Verifying - Enter PEM pass phrase:<MyPEMpassPhrase>

Replace <MyPEMpassPhrase> with your choice of PEM pass phrase, the pass phrase "GlobusTutorial" (whithout the quotes) was used in the compilation of this tutorial.

creating CA config package...done.

A self-signed certificate has been generated

for the Certificate Authority with the subject:

/O=Grid/OU=GlobusTest/OU=ConsortiumTutorial/CN=<MyOrganization>

If this is invalid, rerun this script

/opt/globus-4.0.1/setup/globus/setup-simple-ca

and enter the appropriate fields.

The private key of the CA is stored in

/home/globus/.globus/simpleCA//private/cakey.pem

The public CA certificate is stored in /home/globus/.globus/simpleCA//cacert.pem

The distribution package built for this CA is stored in

/home/globus/.globus/simpleCA//globus_simple_ca_768c46a5_setup-0.19.tar.gz

This file must be distributed to any host wishing to request

certificates from this CA.

CA setup complete.

The following commands will now be run to setup the security

configuration files for this CA:

$GLOBUS_LOCATION/sbin/gpt-build

/home/globus/.globus/simpleCA//globus_simple_ca_768c46a5_setup-0.19.tar.gz

$GLOBUS_LOCATION/sbin/gpt-postinstall

---

setup-ssl-utils: Configuring ssl-utils package

Running setup-ssl-utils-sh-scripts...

Note: To complete setup of the GSI software you need to run the following script as root

to configure your security configuration directory:

/opt/globus-4.0.1/setup/globus_simple_ca_f1f2d5e6_setup/setup-gsi

For further information on using the setup-gsi script, use the -help option. The -default option sets this security configuration to be the default, and -nonroot can be used on systems where root access is not available.

setup-ssl-utils: Complete

- [globus@testX ~]$ /opt/globus-4.0.1/setup/globus_simple_ca_f1f2d5e6_setup/setup-gsi -default -nonroot

setup-gsi: Configuring GSI security

Making trusted certs directory: /opt/globus-4.0.1/share/certificates/

mkdir /opt/globus-4.0.1/share/certificates/

Installing /opt/globus-4.0.1/share/certificates//grid-security.conf.f1f2d5e6...

Running grid-security-config...

Installing Globus CA certificate into trusted CA certificate directory...

Installing Globus CA signing policy into trusted CA certificate directory...

setup-gsi: Complete

การทำ host certificate บนเครื่อง test2

- [root@test2 opt]# export GLOBUS_LOCATION=/opt/globus-4.0.1
- [root@test2 opt]# source /opt/globus-4.0.1/etc/globus-user-env.sh
- [root@test2 opt]# grid-cert-request -host test2.kmitl.ac.th -dir $GLOBUS_LOCATION/etc
- [root@test2 opt]# ls -alh $GLOBUS_LOCATION/etc/hostcert_request.pem

  -rw-r--r-- 1 root root 1.3K Feb 22 10:39 /opt/globus-4.0.1/etc/hostcert_request.pem
- [root@test2 opt]# ls -alh $GLOBUS_LOCATION/etc/hostkey.pem

  -r-------- 1 root root 887 Feb 22 10:39 /opt/globus-4.0.1/etc/hostkey.pem
- [root@test2 opt]# su - globus
- [globus@test2 ~]$ export GLOBUS_LOCATION=/opt/globus-4.0.1
- [globus@test2 ~]$ source $GLOBUS_LOCATION/etc/globus-user-env.sh

- [globus@test2 ~]$ grid-ca-sign -in
$GLOBUS_LOCATION/etc/hostcert_request.pem -out
$GLOBUS_LOCATION/etc/hostcert.pem

ถ้าไฟล์ hostcert.pem **ปรากฏอยู่แล้วให้สั่งคำสั่งตามด้วยตัวเลือก** –f **ซึ่งเป็นการ** Force

- [globus@test2 ~]$ ls -alh $GLOBUS_LOCATION/etc/hostcert.pem
-rw-rw-r-- 1 globus globus 2.5K Feb 22 10:45 /opt/globus-
4.0.1/etc/.hostcert.pem
- [globus@test2 ~]$ ls -alh /home/globus/.globus/simpleCA/newcerts/01.pem
-rw-rw-r-- 1 globus globus 2.5K Feb 22 10:45
/home/globus/.globus/simpleCA/newcerts/01.pem
- [root@test2 opt]# chown root.root /opt/globus-4.0.1/etc/hostcert.pem
- [root@test2 opt]# chmod 644 /opt/globus-4.0.1/etc/hostcert.pem
- [root@test2 opt]# ls -alh /opt/globus-4.0.1/etc/host*.pem
-rw-r--r-- 1 root root 2.5K Feb 22 10:45 /opt/globus-4.0.1/etc/hostcert.pem
-r-------- 1 root root 887 Feb 22 10:39 /opt/globus-4.0.1/etc/hostkey.pem
- [root@test2 opt]# cp /opt/globus-4.0.1/etc/hostcert.pem /opt/globus-
4.0.1/etc/containercert.pem
- [root@test2 opt]# chown globus.globus /opt/globus-4.0.1/etc/containercert.pem
- [root@test2 opt]# ls -alh /opt/globus-4.0.1/etc/containercert.pem
-rw-r--r-- 1 globus globus 2.5K Feb 22 10:57 /opt/globus-
4.0.1/etc/containercert.pem
- [root@test2 opt]# cp /opt/globus-4.0.1/etc/hostkey.pem /opt/globus-
4.0.1/etc/containerkey.pem
- [root@test2 opt]# chown globus.globus /opt/globus-
4.0.1/etc/containerkey.pem
- [root@test2 opt]# ls -alh /opt/globus-4.0.1/etc/containerkey.pem
-r-------- 1 globus globus 887 Feb 22 10:58 /opt/globus-
4.0.1/etc/containerkey.pem

ทำการแก้ไขไฟล์

$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml

โดยเมื่อใช้คำสั่ง cat **จะได้รูปแบบออกมาเป็น ดังนี้**

- [globus@test2 opt]$ cat /opt/globus-4.0.1/etc/globus_wsrf_core/global_security_descriptor.xml

<?xml version="1.0" encoding="UTF-8"?>

<securityConfig xmlns="http://www.globus.org">

<credential>

<key-file value="/opt/globus-4.0.1/etc/containerkey.pem"/>

<cert-file value="/opt/globus-4.0.1/etc/containercert.pem"/>

</credential>

<gridmap value="/opt/globus-4.0.1/etc/grid-mapfile"/>

</securityConfig>

- [globus@test2 ~]$ touch $GLOBUS_LOCATION/etc/grid-mapfile (สร้างกริดแม็พไฟล์)

- [root@test2 opt]# ps auwwwwx|grep postmaster

postgres 26088 0.0 0.3 19476 3172 pts/0 S Feb20 0:00 /usr/bin/postmaster -D /opt/pgsql/data -o -i

- [root@test2 opt]# su – postgres

- -bash-3.00$ createuser -A -d -P globus

Enter password for new user:

Enter it again:

CREATE USER

ทำการแก้ไขไฟล์ /opt/pgsql/data/pg_hba.conf โดยเติมบรรทัดต่อไปนี้ลงไป

host rftDatabase "globus" "161.246.6.113" 255.255.255.0 md5 (ใส่ IP ของเครื่องที่กำลังทำการตั้งค่าระบบ)

จากนั้นใช้คำสั่ง ps –aux แล้วหา process id ของ /usr/bin/postmaster -i -D /opt/pgsql/data แล้วรันคำสั่ง

-bash-3.00$ kill -SIGTERM "process id"

-bash-3.00$ /usr/bin/postmaster -i -D /opt/pgsql/data > /opt/pgsql/logfile 2>&1 &

[root@test2 ~]# su - globus

[globus@test2 ~]$ export GLOBUS_LOCATION=/opt/globus-4.0.1

[globus@test2 ~]$ source $GLOBUS_LOCATION/etc/globus-user-env.sh

- [globus@test2 ~]$ createdb rftDatabase

  CREATE DATABASE

- [globus@test2 ~]$ psql -d rftDatabase -f

  $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql

ผลลัพธ์:

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:6: NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "requestid_pkey" for table "requestid"

CREATE TABLE

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:11: NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "transferid_pkey" for table "transferid"

CREATE TABLE

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:30: NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "request_pkey" for table "request"

CREATE TABLE

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:65: NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "transfer_pkey" for table "transfer"

CREATE TABLE

CREATE TABLE

CREATE TABLE

CREATE INDEX

แก้ไขไฟล์ $GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml โดยแก้ไข password จาก "foo" เป็น password ของ globus user ที่ตั้งไว้ที่ Postgres database จากนั้นที่ user root ทำการแก้ไขไฟล์ /etc/sudoers โดยการเพิ่มบรรทัด ดังต่อไปนี้

globus ALL=(nodeA) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute -g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec/globus-job-manager-script.pl *

globus ALL=(nodeA) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute -g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec/globus-gram-local-proxy-tool *

globus ALL=(nodeB) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute -g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec/globus-job-manager-script.pl *

globus ALL=(nodeB) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute -g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec/globus-gram-local-proxy-tool *

globus ALL=(nodeC) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute -g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec/globus-job-manager-script.pl *

globus ALL=(nodeC) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute -g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec/globus-gram-local-proxy-tool *

ทำการเริ่มการทำงานของคอนเทนเนอร์

- [globus@nodeB ~]$ export JAVA_HOME=/opt/jdk1.5.0_08
- [globus@nodeB ~]$ export PATH=$JAVA_HOME/bin:$PATH
- [globus@nodeB ~]$ which java
  /opt/jdk1.5.0_06/bin/java
- [globus@nodeB ~]$ export GLOBUS_OPTIONS=-Xmx512M
- [globus@nodeB ~]$ $GLOBUS_LOCATION/bin/globus-start-container

ผลลัพธ์:

Starting SOAP server at: https://192.168.31.40:8443/wsrf/services/

With the following services:

[1]: https://192.168.31.40:8443/wsrf/services/TriggerFactoryService

[2]: https://192.168.31.40:8443/wsrf/services/DelegationTestService

[3]: https://192.168.31.40:8443/wsrf/services/SecureCounterService

[4]: https://192.168.31.40:8443/wsrf/services/IndexServiceEntry

[5]: https://192.168.31.40:8443/wsrf/services/DelegationService

[6]: https://192.168.31.40:8443/wsrf/services/InMemoryServiceGroupFactory

[7]: https://192.168.31.40:8443/wsrf/services/mds/test/execsource/IndexService

[8]: https://192.168.31.40:8443/wsrf/services/mds/test/subsource/IndexService

[9]: https://192.168.31.40:8443/wsrf/services/SubscriptionManagerService

[10]: https://192.168.31.40:8443/wsrf/services/TestServiceWrongWSDL

[11]: https://192.168.31.40:8443/wsrf/services/SampleAuthzService

[12]: https://192.168.31.40:8443/wsrf/services/WidgetNotificationService

[13]: https://192.168.31.40:8443/wsrf/services/AdminService

[14]: https://192.168.31.40:8443/wsrf/services/DefaultIndexServiceEntry

[15]: https://192.168.31.40:8443/wsrf/services/CounterService

[16]: https://192.168.31.40:8443/wsrf/services/TestService

[17]: https://192.168.31.40:8443/wsrf/services/InMemoryServiceGroup

[18]: https://192.168.31.40:8443/wsrf/services/SecurityTestService

[19]: https://192.168.31.40:8443/wsrf/services/ContainerRegistryEntryService

[20]: https://192.168.31.40:8443/wsrf/services/NotificationConsumerFactoryService

[21]: https://192.168.31.40:8443/wsrf/services/TestServiceRequest

[22]: https://192.168.31.40:8443/wsrf/services/IndexFactoryService

[23]: https://192.168.31.40:8443/wsrf/services/ReliableFileTransferService

[24]: https://192.168.31.40:8443/wsrf/services/mds/test/subsource/IndexServiceEntry

[25]: https://192.168.31.40:8443/wsrf/services/Version

[26]: https://192.168.31.40:8443/wsrf/services/NotificationConsumerService

[27]: https://192.168.31.40:8443/wsrf/services/IndexService

[28]: https://192.168.31.40:8443/wsrf/services/NotificationTestService

[29]: https://192.168.31.40:8443/wsrf/services/ReliableFileTransferFactoryService

[30]: https://192.168.31.40:8443/wsrf/services/DefaultTriggerServiceEntry

[31]: https://192.168.31.40:8443/wsrf/services/TriggerServiceEntry

[32]: https://192.168.31.40:8443/wsrf/services/PersistenceTestSubscriptionManager

[33]: https://192.168.31.40:8443/wsrf/services/mds/test/execsource/IndexServiceEntry

[34]: https://192.168.31.40:8443/wsrf/services/DefaultTriggerService

[35]: https://192.168.31.40:8443/wsrf/services/TriggerService

[36]: https://192.168.31.40:8443/wsrf/services/gsi/AuthenticationService

[37]: https://192.168.31.40:8443/wsrf/services/TestRPCService

[38]: https://192.168.31.40:8443/wsrf/services/ManagedMultiJobService

[39]: https://192.168.31.40:8443/wsrf/services/RendezvousFactoryService

[40]: https://192.168.31.40:8443/wsrf/services/WidgetService

[41]: https://192.168.31.40:8443/wsrf/services/ManagementService

[42]: https://192.168.31.40:8443/wsrf/services/ManagedExecutableJobService

[43]: https://192.168.31.40:8443/wsrf/services/InMemoryServiceGroupEntry

[44]: https://192.168.31.40:8443/wsrf/services/AuthzCalloutTestService

[45]: https://192.168.31.40:8443/wsrf/services/DelegationFactoryService

[46]: https://192.168.31.40:8443/wsrf/services/DefaultIndexService

[47]: https://192.168.31.40:8443/wsrf/services/ShutdownService

[48]: https://192.168.31.40:8443/wsrf/services/ContainerRegistryService

[49]: https://192.168.31.40:8443/wsrf/services/TestAuthzService

[50]: https://192.168.31.40:8443/wsrf/services/CASService

[51]: https://192.168.31.40:8443/wsrf/services/ManagedJobFactoryService

เริ่มการทำงานของ GridFTP server

- [root@test2 etc]# export GRIDMAP=/opt/globus-4.0.1/etc/grid-mapfile
- [root@test2 etc]# /opt/globus-4.0.1/sbin/globus-gridftp-server -p 2811 -S

หลังจากนั้นจะเริ่มทำการ sign certificate ให้ user จาก host ต่างๆในระบบกริด โดยที่ใช้ในโครงงานจะมีการทำการ sign ให้ user account ต่างๆ ดังนี้

- user nodeA จาก test1
- user nodeB จาก test2
- user nodeC จาก test3

nodeB เป็น user ที่อยู่ที่เครื่อง test2 สามารถทำการ sign certificate ได้เลย แต่ nodeA และ nodeC อยู่ที่เครื่อง test1 และ test3 จะต้องทำการก็อปปี้ตัว simpleCA ที่ทำการสร้างไว้ที่ test2 มาก่อน โดยใช้คำสั่งดังนี้

- [globus@tes1(3) ~]$ scp root@test2:/home/globus/.globus/simpleCA/
  globus_simple_ca_f1f2d5e6_setup-0.19.tar.gz .
  root@test2's password:
  globus_simple_ca_f1f2d5e6_setup-0.19.tar.gz 100% 211KB 210.8KB/s 00:00 :
- [globus@test1(3) ~]$ gpt-build ./globus_simple_ca_f1f2d5e6_setup-0.19.tar.gz

ผลลัพธ์:

gpt-build ====> CHECKING BUILD DEPENDENCIES FOR globus_simple_ca_f1f2d5e6_setup

gpt-build ====> Changing to /home/globus/BUILD/globus_simple_ca_f1f2d5e6_setup-0.19/

gpt-build ====> BUILDING globus_simple_ca_f1f2d5e6_setup

gpt-build ====> Changing to /home/globus/BUILD

gpt-build ====> REMOVING empty package globus_simple_ca_f1f2d5e6_setup-noflavor-data

gpt-build ====> REMOVING empty package globus_simple_ca_f1f2d5e6_setup-noflavor-dev

gpt-build ====> REMOVING empty package globus_simple_ca_f1f2d5e6_setup-noflavor-doc

gpt-build ====> REMOVING empty package globus_simple_ca_f1f2d5e6_setup-noflavor-pgm_static

gpt-build ====> REMOVING empty package globus_simple_ca_f1f2d5e6_setup-noflavor-rtl

- [globus@test1(3) ~]$ gpt-postinstall

running /opt/globus-4.0.1/setup/globus/./setup-ssl-utils.f1f2d5e6..[Changing to /opt/globus-4.0.1/setup/globus/.]

setup-ssl-utils: Configuring ssl-utils package

Running setup-ssl-utils-sh-scripts...

Note: To complete setup of the GSI software you need to run the following script as root to configure your security configuration directory:

/opt/globus-4.0.1/setup/globus_simple_ca_f1f2d5e6_setup/setup-gsi

For further information on using the setup-gsi script, use the -help option. The -default option sets this security configuration to be the default, and -nonroot can be used on systems where root access is not available.

setup-ssl-utils: Complete

..Done

- [globus@test1(3) ~]$ /opt/globus-4.0.1/setup/globus_simple_ca_f1f2d5e6_setup/setup-gsi -default -nonroot

setup-gsi: Configuring GSI security

Making trusted certs directory: /opt/globus-4.0.1/share/certificates/

mkdir /opt/globus-4.0.1/share/certificates/

Installing /opt/globus-4.0.1/share/certificates//grid-security.conf.f1f2d5e6...

Running grid-security-config...

Installing Globus CA certificate into trusted CA certificate directory...

Installing Globus CA signing policy into trusted CA certificate directory...

setup-gsi: Complete

จากนั้นทำการ sign certificate ให้ user

- [nodeX@testX ~]$ export GLOBUS_LOCATION=/opt/globus-4.0.1
- [nodeX@testX ~]$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
- [nodeX@testX ~]$ grid-cert-request
- [nodeX@testX ~]$ ls -l .globus/usercert_request.pem

  -rw-r--r-- 1 nodeX users 1346 Feb 23 10:41 .globus/usercert_request.pem
- [globus@testX ~]$ cd .globus/simpleCA/
- [globus@testX simpleCA]$ scp

  root@testX:/home/nodeX/.globus/usercert_request.pem .

  root@testX's password:

  usercert_request.pem 100% 1346 1.3KB/s 00:00
- [globus@testX simpleCA]$ grid-ca-sign -in ./usercert_request.pem -out ./usercert.pem

ผลลัพธ์:

To sign the request please enter the password for the CA key:

The new signed certificate is at: /home/globus/.globus/simpleCA//newcerts/02.pem

- [nodeX@testX ~]$ scp root@test2:/home/globus/.globus/simpleCA/usercert.pem $HOME/.globus/usercert.pem
- [nodeX@testX ~]$ ls -alh .globus/usercert.pem

  -rw-r--r-- 1 nodeX users 2.5K Feb 23 11:39 .globus/usercert.pem

หลังจากนั้นให้ทำการ sign host certificate ซึ่งอยู่ในขั้นตอนที่ได้กล่าวมาแล้ว หลังจากการลงระบบเสร็จสิ้น ถ้าต้องการ test ระบบจะมีวิธีทำเพิ่มเติมเข้าไปดูได้ที่

http://www.globusconsortium.org/tutorial

ภาคผนวก ค.

ขั้นตอนการลงระบบ TORQUE-2.0.0p7

Deploying Torque (OpenPBS)

Deplaying RSH

ที่user root เราจะเริ่มต้นด้วยการตรวจสอบว่าตัว xinetd ได้ถูกติดตั้งหรือยัง

```
[root@testX data]# rpm -qa|grep xinetd
xinetd-2.3.13-6
```

จากนั้นจะทำการตรวจสอบว่าxinetdนั้นถูกตั้งค่าให้เริ่มทำงานทุกครั้งที่เปิดเครื่องหรือไม่

```
/sbin/chkconfig --list | grep xinetd
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
xinetd based services:
```

ถ้าหากยังไม่ได้ลง xinetd สามารถติดตั้งได้โดยใช้คำสั่ง

```
yum install xinetd
```

จากนั้นจะทำการลงแพ็กเกจของ rsh ที่จำเป็นอีก 2 ตัว

```
yum install rsh-server
yum install rsh
```

ตามปกติแล้ว บริการของ rsh และ rlogin จะยังไม่สามารถใช้งานได้ เราจำเป็นที่จะต้องไปแก้ไขไฟล์เหล่านี้เสียก่อน

```
/etc/xinetd.d/rsh
/etc/xinetd.d/rlogin
```

และให้เปลี่ยน disable ให้ค่าเป็น no

จากนั้นให้ทำคำสั่ง

```
/etc/init.d/xinetd restart
```

เพื่อรีสตาร์ท xinetd (หรือใช้คำสั่ง start หากxinetd ยังไม่ได้รันอยู่)

จากนั้นเราจะทำการตั้งค่าให้เครื่องสามารถใช้งาน rsh และ rlogin ได้เฉพาะเครื่องนี้เท่านั้น โดยการแก้ไขไฟล์ /etc/hosts.equiv และเพิ่ม IP ของ test2 ไป 1 บรรทัดหากเราต้องการตรวจสอบว่าเราพิมพ์ IP ถูกต้องหรือไม่ให้เราใช้คำสั่ง

```
[root@test2 xinetd.d]# cat /etc/hosts.equiv
192.168.31.40
```

ต่อไปก็ทดสอบว่า rsh ทำงานได้ถูกต้องหรือไม่สำหรับ user 'nodeX'

```
[nodeX@testX ~]$ /usr/bin/rsh nodeb /usr/bin/whoami
NodeX
```

Downloading and Installing Torque

สำหรับเดโมตัวนี้ ตัวtorque.tar.gz นั้นจะถูกโหลดไปยังไดเร็กทอรี่ของ systemuser home (/home/systemuser) และทำการคลายไฟล์ , ตั้งค่า และการลงโปรแกรมจะทำภายใต้ user 'root' โดยในขั้นตอนแรกเราจะทำการโหลดตัวโปรแกรมโดยใช้คำสั่ง 'wget'

wget http://clusterresources.com/downloads/torque/torque-2.0.0p7.tar.gz

จากนั้นให้พิมพ์คำสั่งดังต่อไปนี้ เพื่อทำการคลายไฟล์ ตั้งค่า สร้าง และติดตั้ง Torque(Open PBS)

```
tar -zxf torque-2.0.0p7.tar.gz
cd torque-2.0.0p7
./configure --prefix=/opt/pbs
make
make install
make packages
./torque-package-clients-linux-i686.sh --install --destdir /opt/pbs
./torque-package-mom-linux-i686.sh --install --destdir /opt/pbs
```

Configuring and Deploying Torque (PBS)

ที่ root ให้พิมพ์ตามคำสั่งนี้เพื่อเริ่มต้นตั้งค่าให้กับ PBS server (ทำเฉพาะเครื่องที่เป็น PBS server)

```
/opt/pbs/sbin/pbs_server -t create
```

จากนั้นให้รันคำสั่งต่อไปนี้ที่ root

```
/opt/pbs/bin/qmgr
```

หลังจากรันคำสั่ง qmgr แล้ว จะเป็นการสตาร์ท "prompt session" ซึ่งจะเป็นส่วนที่ให้เราพิมพ์คำสั่งที่ใช้ในการตั้งค่าของ PBS โดยให้เราพิมพ์ตั้งค่าตามนี้ โดยการตั้งชื่อของโฮสต์ในคำสั่งแรกให้ใช้เป็นตัวอักษรตัวเล็กทั้งหมด ยกเว้นว่าชื่อของโฮสต์จะมีตัวอักษรตัวใหญ่

```
Qmgr: set server operators = root@test2.kmitl.ac.th
Qmgr: create queue batch
Qmgr: set queue batch queue_type = Execution
Qmgr: set queue batch started = True
Qmgr: set queue batch enabled = True
Qmgr: set server default_queue = batch
```

```
Qmgr: set server resources_default.nodes = 1
Qmgr: set server scheduling = True
Qmgr: quit
```

ต่อจากนั้นเราจะทำการตั้งค่าให้ PBS รู้จักว่ามีโหนดไหนบ้างในคลัสเตอร์ที่สามารถใช้ได้บ้าง โดยการใช้user 'root' ในการสร้างไฟล์ /usr/spool/PBS/server_priv/nodes

```
touch /usr/spool/PBS/server_priv/nodes
```

แล้วใช้โปรแกรม VI หรือ editor ตัวอื่นในการเพิ่มบรรทัดดังนี้

```
testX.kmitl.ac.th np=2
```

หมายเหตุ 'np=2' นั้นเกิดจากการที่เครื่องทดลองที่ใช้ในตัวอย่างนี้เป็นแบบที่มี 2 หน่วยประมวลผล(2 processors) หากเครื่องของคุณที่ใช้มีเพียงหน่วยประมวลผลเดียว ไม่จำเป็นที่จะต้องตั้งค่า

แล้วลองใช้คำสั่ง cat เพื่อดูค่าที่พิมพ์ในไฟล์

```
[root@testX torque-2.0.0p7]# cat /usr/spool/PBS/server_priv/nodes
testX.kmitl.ac.th np=2
```

ต่อจากนั้นเราจะทำการตั้งค่า PBS เพื่อให้รู้ว่าโหนดใดในคลัสเตอร์ที่เป็นเซิร์ฟเวอร์ โดยการสร้างไฟล์ /usr/spool/PBS/mom_priv/jobs/config และแก้ไขไฟล์ให้เป็นไปตามนี้

```
[root@testX mom_priv]# cat /usr/spool/PBS/mom_priv/jobs/config
$pbsserver    nodeB.ps.univa.com
$logevent    255
```

Start PBS

รันคำสั่ง 3 คำสั่งนี้ด้วย user 'root' เพื่อสตาร์ทองค์ประกอบต่างๆของ PBS (เครื่องโหนดอื่นที่ไม่ใช่ server ไม่จำเป็นต้องรันคำสั่งสุดท้าย)

```
[root@testX ]# /opt/pbs/sbin/pbs_mom
[root@testX ]# /opt/pbs/bin/qterm -t quick
[root@testX ]# /opt/pbs/sbin/pbs_server
```

หลังจากสตาร์ททุกคำสั่งเรียบร้อยแล้ว เราจะสามารถ query เพื่อดูโหนดต่างๆในคลัสเตอร์ได้

```
[root@nodeB mom_priv]# /opt/pbs/bin/pbsnodes -a
nodeB.ps.univa.com
state = free
```

np = 2

ntype = cluster

status = opsys=linux,uname=Linux nodeB.ps.univa.com 2.6.11-1.1369_FC4smp #1 SMP Thu Jun 2 23:08:39 EDT 2005 i686, sessions=1894, nsessions=1, nusers=1, idletime=7528, totmem=3065064kb, availmem=3015308kb, physmem=1033456kb, ncpus=2,loadave=0.00, netload=127730552, state=free, jobs=? 0, rectime=1139525328

พอถึงจุดนี้ PBSพร้อมทีจะรับงานแล้ว แต่มันยังไม่สามารถรันงานได้เพราะยังไม่มีschedulerมาจัดตารางงาน แต่ในตัวอย่างนี้เราจะใช้ scheduler ที่มีมาให้อยู่แล้ว โดยทำการสตาร์ท simple scheduler ด้วยuser 'root'

/opt/pbs/sbin/pbs_sched

Testing PBS

ตอนนี้ทุกๆ user บนโหนด test2 จะสามารถส่งงานไปยัง PBS ได้ และสามารถ query ดูสถานะของงานได้ ในการทดลองนี้เราจะใช้ user ' nodeB' เพื่อรันคำสั่งต่อไปนี้ เพื่อแบ่งงานที่จะพักการทำงานของคอมพิวเตอร์ 1 นาที แล้วพิมพ์ผลลัพธ์ ออกมาเป็นเวลาและวันที่ ณ ปัจจุบัน

[jane@nodeB ~]$ echo "sleep 60;date" | /opt/pbs/bin/qsub

คุณจะเห็นผลลัพธ์ที่ได้ออกมาเป็น

4.test2.kmitl.ac.th

หมายเหตุ หมายเลข 4 นั้นอาจเป็นตัวเลขอื่นได้ ขึ้นอยู่กับจำนวนครั้งที่คุณรันคำสั่งนี้ โดยตัวนี้จะเป็นเพียงแค่ ID ของงานตัวอย่างเท่านั้น

ในการขอดูสถานะของงาน เราจะรันคำสั่ง

[nodeB@test2 ~]$ /opt/pbs/bin/qstat

Job id	Name	User	Time Use	S	Queue
4.test2	STDIN	nodeB	0	R	batch

หลังจากที่งานเสร็จแล้ว เราสามารถดูไฟล์ผลลัพธ์ได้โดย

[jane@nodeB ~]$ ls

STDIN.e4 STDIN.o4

ไฟล์ "STDIN.o4"จะบรรจุผลลัพธ์ที่ได้จากการรันงาน ซึ่งจะแสดงเป็นเวลาและวันที่

[jane@nodeB ~]$ cat STDIN.o4

Thu Feb 9 17:09:53 CST 2006