การประมวลผลแบบกริด

GRID COMPUTING

นางสาว ณีรนุช ลิปิพัฒนากุล
นาย ทศพร คชานุบาล
นางสาว สุรีพล องคนิกูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2548

การประมวลผลแบบกริด

GRID COMPUTING

โดย

นางสาว ณีรนุช ลิปิพัฒนากุล

นาย ทศพร คชานุบาล

นางสาว สุรีพล องคนิกูล

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง (ชื่อภาษาไทย) การประมวลผลแบบกริด

(ชื่อภาษาอังกฤษ) GRID COMPUTING

ผู้จัดทำ

1. นางสาว ณีรนุช ลิปีพัฒนากุล รหัสนักศึกษา 45010258
2. นาย ทศพร คชานุบาล รหัสนักศึกษา 45010297
3. นางสาว สุรีพร องคนิกูล รหัสนักศึกษา 45010885

อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การประมวลผลแบบกริด

นางสาว ฉีรนุช ลิปิพัฒนากุล 45010258
นาย ทศพร คชานุบาล 45010297
นางสาว สุรีพร องคนิกูล 45010885
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2548

บทคัดย่อ

ปัจจุบันเทคโนโลยีด้านการติดต่อสื่อสารและเครื่องคอมพิวเตอร์ส่วนบุคคลได้พัฒนาไปอย่างรวดเร็ว ดังนั้นจึงเป็นโอกาสที่ดีที่จะนำคอมพิวเตอร์เหล่านี้มาใช้ประโยชน์แทนที่ ซุปเปอร์คอมพิวเตอร์ที่มีต้นทุนสูงและยากแก่การบำรุงรักษา นักวิชาการจึงพัฒนาเทคโนโลยีใหม่ที่เรียกว่ากริด ("Grid") ขึ้นมาใช้ในการประมวลผลงานใหญ่ๆ ซึ่งกริดนี้ได้นำเอาข้อดีของคลัสเตอร์ (Cluster) และ เพียร์ทูเพียร์ (Peer-to-Peer) รวมเข้าไว้ด้วยกัน กล่าวคือ คอมพิวเตอร์แต่ละตัวร่วมกันประมวลผลงานชิ้นเดียวกัน ซึ่งคล้ายกับ Cluster แต่ไม่ต้องมีศูนย์กลางที่ใช้ในการแจกจ่ายงาน ซึ่งคล้ายกับเพียร์ทูเพียร์รวมถึงข้อดีที่ว่ากริดไม่จำเป็นที่จะต้องอยู่ในเครือข่ายภายใน(LAN)เดียวกัน กริดจะใช้อินเตอร์เน็ตเป็นเครือข่ายที่ใช้รองรับการประมวลผล ดังนั้นไม่ว่าจะอยู่มุมใดของโลกก็สามารถเป็นส่วนหนึ่งของกริดได้ ซึ่งในรายงานฉบับนี้จะนำเสนอสถาปัตยกรรม รายละเอียดการทำงาน รูปแบบ และเครื่องมือที่ใช้ของกริดตลอดจนมาตรฐานที่ใช้ทั้งหมด

GRID COMPUTING

Miss	Neeranuch	Lipipattanakul	45010258
Mr.	Tossaporn	Kachanubarn	45010297
Miss	Sureeporn	Ongkanigool	45010885
Dr.	Voravat	Limpoka	Advisor

Academic Year 2004

ABSTRACT

Nowadays the communication technology and personal computer have been rapidly developed. Therefore it is a great opportunity to use these utilization of computers instead of super computer, which has higher cost and difficult to maintain. So the researchers have developed the new technology called "Grid" for computing the large project . This Grid is the combination of the advantage of Cluster and Peer-to-Peer. Each computer works on the same project at the same time but in the different section of the project which is similar to Cluster but there is no center for distributing job which is similar to Peer-to-Peer. The advantage of Grid is it does not have to be in the same Local Area so internet or the other network is used. these networks are for supporting the Grid processing so wherever the user is in the world, he/she can be part of Grid. This project presents the architecture, function, pattern, tools, and the standard that is used. Also it's include Parallel Program implementation.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก.ดร.วรวัฒน์ ลิ้มโภคา ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากท่านอาจารย์และขอขอบพระคุณเป็นอย่างสูง

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

ขอขอบคุณบัณฑิตศึกษาและบัณฑิตวิทยาลัย คณะวิศวกรรมศาสตร์ที่ให้ความช่วยเหลือ ในเรื่องต่างๆ

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

ณีรนุช ลิปิพัฒนากุล
ทศพร คชานุบาล
สุรีพร องคนิกูล

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1

บทนำ

ปัจจุบันเทคโนโลยีด้านการสื่อสารผ่านระบบเครือข่ายเน็ตเวิร์คได้พัฒนาขึ้นกว่าเดิมมากรวมถึงความสามารถของคอมพิวเตอร์ส่วนบุคคล ได้พัฒนาไปอย่างรวดเร็วทั้งในเรื่องของความเร็วและจำนวนพื้นที่ในการเก็บข้อมูล จึงเป็นโอกาสที่ดีที่จะนำคอมพิวเตอร์เหล่านี้มาใช้ประโยชน์แทนที่ ซุปเปอร์คอมพิวเตอร์(super computer) ที่มีราคาสูงและยากแก่การบำรุงรักษา นักวิชาการจึงพัฒนาเทคโนโลยีใหม่ที่เรียกว่า "กริด (Grid)" ขึ้นมาเพื่อใช้ในการประมวลผลงานใหญ่ ได้นำเอาข้อดีของคลัสเตอร์ (Cluster) และเพียร์ทูเพียร์ (Peer-to-Peer) รวมเข้าไว้ด้วยกัน กล่าวคือ คอมพิวเตอร์แต่ละตัวสามารถร่วมกันประมวลผลงานชิ้นเดียวกัน ซึ่งคล้ายกับคลัสเตอร์ แต่ไม่จำเป็นต้องมีศูนย์กลางที่ใช้ในการแจกจ่ายงาน ซึ่งคล้ายกับเพียร์ทูเพียร์ (Peer-to-Peer) ทำให้สามารถใช้งานได้อย่างยืดหยุ่น จากข้อดีของระบบ กริดที่กล่าวมา ทำให้คณะผู้จัดทำโครงงานเกิดแนวคิดที่จะออกแบบและสร้างระบบกริดขึ้นเพื่อใช้งาน เรียนรู้และทดสอบความสามารถของระบบกริด โดยเลือกใช้ Globus Toolkit 3.2.1 เป็น มิดเดิ้ลแวร์(middleware)ในการพัฒนาระบบ และทดลองพัฒนาโปรแกรมที่สามารถใช้งานบนกริดได้ โดยเลือกพัฒนาโปรแกรมการคำนวณเมตริก เนื่องจากสามารถทำความเข้าใจการทำงานได้ง่าย สามารถออกแบบรูปแบบของการกระจายงานได้และเห็นโครงสร้างของการทำงานได้อย่างชัดเจน เพื่อนำไปเป็นพื้นฐานของการพัฒนาโปรแกรมอื่นๆที่มีความซับซ้อนมากๆได้

1.1.วัตถุประสงค์ของโครงงาน

1.1.1. เพื่อศึกษาแนวคิดและกระบวนการทำงานของการประมวลผลแบบกริด

1.1.2. เพื่อศึกษาการสร้างระบบกริดเพื่อการใช้งานจริง

1.1.3. เพื่อศึกษาเครื่องมือที่จะนำมาใช้เป็นมิดเดิ้ลแวร์(middleware)ตลอดจนวิธีการใช้งานเครื่องมือนี้

1.1.4. เพื่อศึกษาการเขียนโปรแกรมแบบขนาน

1.1.5. เพื่อศึกษาและออกแบบแอพพลิเคชั่นที่เหมาะสมในการใช้งานในระบบกริดได้

1.1.6. ทำการทดลอง และศึกษาถึงประสิทธิภาพการทำงาน วัดผล และเปรียบเทียบระหว่างการทำงานบนเครื่องเดียว กับการทำงานแบบกระจาย

1.2. ขอบเขตของโครงงาน

ในปริญญานิพนธ์ฉบับนี้ได้นำเสนอวิธีการประมวลผลแบบกริด ซึ่งใช้มิดเดิ้ลแวร์ชื่อ globus ในการติดต่อระหว่างฮาร์ดแวร์และซอฟแวร์ โดยเขียนโปรแกรมกระจายงานไปยังเครื่องไคล์เอนท์ในระบบ

ในลักษณะการประมวลผลแบบขนาน โดยแอพพลิเคชั่นที่เลือกใช้ในปริญญานิพนธ์ฉบับนี้คือ การคำนวณเมตริก

1.3.วิธีการดำเนินงาน

1.3.1. ศึกษาค้นคว้าทฤษฎีเกี่ยวกับการประมวลผลแบบกริด

1.3.2. ศึกษาเทคโนโลยีที่เกี่ยวข้องในการประมวลผลแบบกริด

1.3.3. ศึกษาวิธีการใช้งาน Globus Toolkit (GT3.2.1) รวมถึงองค์ประกอบภายในเครื่องมือ

1.3.4. สร้างระบบการประมวลผลแบบกริดบนระบบปฏิบัติการ Linux Redhat 9

1.3.5. ติดตั้งและเตรียมสภาพแวดล้อมของซอฟแวร์ที่ต้องใช้ร่วมกับ Globus Toolkit เพื่อให้แอพพลิเคชั่นที่ต้องการพัฒนาสามารถทำงานได้

1.3.6. ศึกษาถึงการประยุกต์ใช้และการพัฒนาระบบประมวลผลแบบกริดในปัจจุบัน

1.3.7. ออกแบบแอพพลิเคชั่นที่เหมาะสมกับการประมวลผลแบบกริด

1.3.8. ทดลองเขียนแอพพลิเคชั่นเบื้องต้นที่สามารถใช้งานบนระบบประมวลผลแบบกริดที่สร้างขึ้นได้

1.3.9. ทดสอบการทำงานของแอพพลิเคชั่นที่สร้างขึ้น

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.เนื้อหาของรายงาน

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 10 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 ถึง 4 กล่าวถึงพื้นฐานความรู้ที่ต้องใช้ในโครงงาน อันได้แก่ แนวความคิดของระบบกริดและระบบอื่นๆที่เป็นแรงจูงใจในการพัฒนากริด และอธิบายถึงองค์ประกอบในเครื่องมือที่ใช้ในการทดลอง (globus toolkit)

บทที่ 5 กล่าวถึงแนวคิด และวิธีการเขียนโปรแกรมแบบขนานที่จะใช้ในการทดลอง

บทที่ 6 ถึง 8 กล่าวถึงเทคโนโลยีที่ใช้ในการทดลอง และวิธีการเขียนเซอร์วิสเพื่อเรียกใช้ในระบบ

บทที่ 9 กล่าวถึงลำดับขั้นตอนในการทดลองและผลการทดลอง การหาค่าสมรรถนะของระบบ การวัดประสิทธิภาพของระบบ พารามิเตอร์ที่ใช้และผลที่ได้จากการจำลองระบบ ผลการทดลองหรือผลการทำงานทั้งหมด โดยการสรุปเป็นกราฟ

บทที่ 10 เป็นบทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 10
บทวิจารณ์และสรุป

จากการทดลองใช้งาน กริดเซอร์วิสบนระบบกริดที่สร้างขึ้น ทำให้สังเกตได้ว่าระบบกริดเป็นระบบที่มีความยืดหยุ่นมากในแง่ของการเป็น distributed system คือสามารถกระจายงานไปในหลายๆ เครื่องได้โดยไม่จำกัดการกระจายงานว่าจะต้องใช้กี่เครื่อง และยังสามารถลดเวลาของการคำนวณตามจำนวนเครื่องที่สามารถเรียกใช้เซอร์วิสได้ โดยที่เมื่อมีเครื่องที่สามารถรับงานย่อยไปใช้ประมวลผลหลายๆเครื่อง จะทำให้ใช้เวลาในการคำนวณโดยรวมน้อยลง และเครื่องที่รับงานไปช่วยประมวลผลก็ยังสามารถใช้งานได้ตาม ปกติ

10.1 สรุปผลการทดลอง

จากผลการทดลองสามารถสรุปได้ดังนี้

- การใช้จำนวนเครื่องคอมพิวเตอร์หลายๆเครื่องร่วมกันคำนวณ จะทำให้ความเร็วโดยรวมเร็วกว่าการประมวลผลในเครื่องเดียว โดยอาศัยวิธีการแบ่งงานออกเป็นงานย่อยแล้วกระจายไปยังเครื่องอื่นให้ช่วยกันประมวลผล
- การใช้งานกับการคำนวณใหญ่ๆ จะเห็นผลลัพธ์จากการใช้งานระบบกริดได้ชัดเจนกว่า เช่น การคำนวณเมตริกขนาด 320*320 จะเห็นความแตกต่างระหว่างการคำนวณบนเครื่องเดียวกับการกระจายไปหลายๆเครื่อง ได้ชัดเจนกว่าการคำนวณมตริกขนาด 10*10
- การเพิ่มจำนวน thread ทำให้สามารถทำงานได้เร็วขึ้น แต่ก็ขึ้นอยู่กับขนาดของงานด้วย โดยถ้าขนาดของงานเล็กถึงแม้ว่าจะใช้จำนวน thread มากก็ไม่เกิดผลลัพธ์ที่ดีขึ้น

10.2 ข้อจำกัดของการใช้งานระบบ

- การแบ่งงานที่ย่อยเกินไป จะทำให้เสียโอเวอร์เฮดในการส่งข้อมูลมาก ดังนั้นจึงไม่เหมาะสมที่จะใช้ในการคำนวณที่มีรายละเอียดน้อย
- หากงานย่อยแต่ละงานไม่เกี่ยวข้องกันหรือไม่ต้องรอผลลัพธ์ของกันและกันจะทำให้ประสิทธิภาพโดยรวมของการใช้งานระบบมีมากขึ้น
- เวลาที่ใช้ในการคำนวณแต่ละงานอาจไม่คงที่ เนื่องจากความเร็วจะขึ้นอยู่กับความคับคั่งของการใช้งานเน็ตเวิร์คและโหลดของการใช้งานเครื่องที่เป็น service host
- หากมีการใช้งานเครื่องที่เป็น service host พร้อมกันด้วย จะทำให้ความเร็วในการคำนวณโดยรวมตกลง

- อาจปัญหาเกี่ยวกับพื้นที่ swap เนื่องจากการกระจายงานใช้การแบ่ง thread ซึ่งจำเป็นต้องใช้หน่วยความจำชั่วคราวเยอะ หากมีการแบ่ง thread ที่มากเกินไปอาจทำให้เกิดปัญหา stack overflow ได้
- การกระจายงานจะใช้หลักการของการเขียนโปรแกรมแบบขนานซึ่งยังไม่แพร่หลายนัก อาจเกิดปัญหาในการนำไปพัฒนาต่อได้

10.3 แนวทางในการพัฒนาต่อ

- ทำการพัฒนาในส่วนของแอพพลิเคชั่นที่ใช้ในการแสดงความสามารถของกริดให้เห็นผลที่ชัดเจนขึ้น
- ออกแบบส่วนติดต่อกับผู้ใช้งานให้ผู้ใช้งานสามารถใช้งานได้ง่ายและสะดวกขึ้น
- พัฒนาในส่วนของอัลกอริทึมในการแตกงานให้เหมาะสมกับอุปกรณ์ที่ใช้งาน
- นำแอพพลิเคชั่นในด้านอื่นๆมาผสมผสานกับความสามารถในการประมวลผลของระบบกริด

บรรณานุกรม

[1] Globus Alliance, http://www.globus.org

[2] Java, http://java.sun.com

[3] Apache Ant, http://ant.apache.org

[4] Junit, http://www.junit.org

[5] What is the Grid, http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf

[6] Gestalt of the Grid, http://www.mcs.anl.gov/~gregor/ papers/vonLaszewski--gestalt.pdf

[7] The Anatomy of the Grid: Enabling Scalable Virtual Organizations, http://www.globus.org/research/papers/anatomy.pdf

[8] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, http://www.globus.org/research/papers/ogsa.pdf

[9] Gt3 Core, http://www-unix.globus.org/toolkit/3.0/ogsa/docs/gt3_core.pdf

[10] Globus Toolkit 3 Quick Start, http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf

[11] The Globus Toolkit 3 Programmer's Tutorial, http://www.casa-sotomayor.net/gt3-tutorial/

[12] How to install Glubus and write a hello world service, http://www.dutchgrid.nl/install/gang/ HowTo-GlobusInstallationExperience.pdf

[13] Implementing a Distributed Master/Slave Grid Service with Globus Toolkit 3 (GT3), http://dps.uibk.ac.at/~gregor/mandel.pdf

[14] Ian Foster, http://www-fp.mcs.anl.gov/~foster/

[15] IBM Redbook, http://www.ibm.com/redbooks/

บทที่ 2
การประมวลผลแบบกริด

2.1. แนวความคิดของการประมวลผลแบบกริด

การประมวลผลแบบกริดเป็นการประมวลผลแบบกระจายที่มีจุดประสงค์เพื่อที่จะสร้างคอมพิวเตอร์เสมือนที่มีขนาดใหญ่ และมีกำลังในการประมวลผลมากโดยใช้การรวมกลุ่มของระบบที่มีความแตกต่างกันซึ่งเชื่อมต่อกันผ่านทางเครือข่ายคอมพิวเตอร์เพื่อแชร์ทรัพยากรที่มีความหลากหลาย เพื่อพัฒนาไปเรื่อยๆ จนเข้าใกล้การคำนวณแบบกระจายที่รวบรวมทรัพยากรต่างๆเพื่อให้ผู้ใช้เข้าถึงได้อย่างไม่จำกัด ในปัจจุบันการใช้การประมวลผลแบบกริดมีจุดประสงค์หลัก 2 ประการ ประการแรกคือ เพื่อการวิจัย และทดลอง ประการที่สอง คือ เพื่อพัฒนาไว้ใช้งานในองค์กร

การประมวลผลแบบกริด ทำให้สามารถใช้งานทรัพยากรของเครื่องคอมพิวเตอร์ที่มีอยู่ได้อย่างเต็มประสิทธิภาพและคุ้มค่า โดยการประมวลผลแบบกริดกำลังดำเนินรอยตาม SETI@home, Beowulf, Condor และ Unicore ในการสร้างทรัพยากรขนาดใหญ่เพื่อใช้ในการประมวลผล คือทำให้เครือข่ายกลายเป็นเครื่องคอมพิวเตอร์ขนาดยักษ์เพียงเครื่องเดียวที่ทุกคนสามารถเข้ามาใช้งานได้

การประมวลผลแบบกริดมีความคล้ายคลึงการใช้พลังงานไฟฟ้าคือเมื่อคุณเสียบปลั๊กเครื่องใช้ไฟฟ้าสิ่งที่คุณคาดหวังจะได้รับคือพลังงานที่มีค่าโวลท์ถูกต้อง แต่ไม่มีความจำเป็นต้องทราบแหล่งที่มาของพลังงานนั้น โดยบริษัทสาธารณูปโภคท้องถิ่นของคุณจะทำการเชื่อมต่อไปยังเครือข่ายของเครื่องกำเนิดไฟฟ้าและแหล่งที่มาของพลังงานตลอดจนจัดการคุณภาพการบริการเพื่อตอบสนองความต้องการพลังงานของคุณ มุมมองของการประมวลผลแบบกริดก็เหมือนกัน คือมีการจัดโครงสร้างที่เหมาะสมและผู้ใช้จะเข้าถึงคอมพิวเตอร์เสมือนที่เชื่อถือได้และเหมาะกับความต้องการของผู้ใช้ คอมพิวเตอร์เสมือนจะประกอบด้วยทรัพยากรในการประมวลผลที่แตกต่างกันจำนวนมาก แต่ผู้ใช้จะไม่สามารถมองเห็นทรัพยากรแต่ละอันเหล่านี้ เหมือนกับที่ผู้อุปโภคพลังงานไฟฟ้าจะไม่ทราบว่าพลังงานที่ใช้อยู่นั้นถูกผลิตได้อย่างไรซึ่งในการประมวลผลแบบกริดย่อมต้องมีมาตรฐานเพื่อให้มีโครงสร้างที่ปลอดภัยและทนทาน ตัวอย่างของมาตรฐานที่กล่าวถึงนี้ เช่น Open Grid Service Architecture (OGSA) และเครื่องมือที่เป็นที่นิยม เช่น Globus Toolkit ซึ่งให้ เฟรมเวิร์ค ที่จำเป็น

2.2. ทฤษฎีที่เกี่ยวข้อง

2.2.1. ซุปเปอร์คอมพิวเตอร์ (Supercomputer)

เมื่อคอมพิวเตอร์เข้ามามีบทบาทในการทำงานมากขึ้นจึงเกิดความต้องการคอมพิวเตอร์ที่มีกำลังในการประมวลผลมากๆ ซึ่งเป็นที่มาของซุปเปอร์คอมพิวเตอร์

ซุปเปอร์คอมพิวเตอร์เป็นเครื่องคอมพิวเตอร์ที่เหมาะกับงานคำนวณที่ต้องมีการคำนวณตัวเลขจำนวนหลายล้านตัวภายในเวลาอันรวดเร็ว เช่น งานพยากรณ์อากาศที่ต้องนำข้อมูลต่างๆ เกี่ยวกับอากาศทั้งระดับภาคพื้นดินและระดับชั้นบรรยากาศเพื่อดูการเคลื่อนไหวและการเปลี่ยนแปลงอากาศซึ่งจำเป็นต้องใช้เครื่องคอมพิวเตอร์ที่มีสมรรถนะสูงมาก นอกจากนี้ยังมีงานอื่นๆอีกเป็นจำนวน มากที่ต้องใช้ซุปเปอร์คอมพิวเตอร์ที่มีความเร็วสูง เช่น งานควบคุมขีปนาวุธ, งานควบคุมทางอวกาศ, งานประมวลผลภาพทางการแพทย์, งานด้านวิทยาศาสตร์ โดยเฉพาะทางด้านเคมี เภสัชวิทยา และงานด้านวิศวกรรมการออกแบบซุปเปอร์คอมพิวเตอร์สามารถทำงานได้เร็วเพราะมีโครงสร้างการคำนวณพิเศษ

2.2.1.1. Multiprocessor

คือคอมพิวเตอร์หนึ่งเครื่องมีซีพียูหลายตัว ซีพียูแต่ละตัวช่วยกันทำงานซึ่งได้แก่ SMP (Symmetric Multiprocessing)คือโปรเซสเซอร์(processor)หลายตัวใช้ทรัพยากรของระบบเช่น บัส(bus), หน่วย ความจำ, อินพุท เอาท์พุท(I/O) ร่วมกันโดยไม่สามารถแบ่งแยกเป็นส่วนย่อยๆ ได้ โดย SMP มีคุณสมบัติดังนี้

- มีความสามารถเทียบได้กับโปรเซสเซอร์ 2 ตัวหรือมากกว่านั้น
- มีการแบ่งหน่วยความจำหลักออกเป็นส่วนๆและติดต่อกับ อินพุท เอาท์พุท ได้ง่ายด้วยการเชื่อมต่อโดยใช้ระบบบัสหรือการเชื่อมต่อปกติ ดังนั้นจึงใช้เวลาในการเข้าถึงข้อมูลเท่ากับ โปรเซสเซอร์เพียงตัวเดียว
- คอมพิวเตอร์ทุกเครื่องสามารถทำงานได้ด้วยตัวเอง
- ระบบทุกระบบจะทำงานร่วมกันโดยมีระบบควบคุมระบบหนึ่งที่ควบคุมการทำงานระหว่างโปรเซสเซอร์แต่ละตัว

2.2.1.2. MPP (Massively Parallel Processor)

คือ คอมพิวเตอร์ที่มีโปรเซสเซอร์หลายตัวโดยจะทำการแบ่งแยกงานให้กับตัวประมวลผลแต่ละตัวซึ่งมีทรัพยากรอันได้แก่ อินพุท เอาท์พุท, หน่วยความจำ เป็นของตนเองและจะทำงานขนานกันไป

2.2.1.3. HPC (High Performance Computer)

คือคอมพิวเตอร์มีซีพียูและอุปกรณ์อื่นๆจำนวนมากเพื่อให้สามารถทำงานที่ต้องทำการคำนวณตัวเลขจำนวนมากๆได้เร็วขึ้น เช่น งานพยากรณ์อากาศเป็นต้น

2.2.2. เพียร์ทูเพียร์ (Peer-to-Peer ,P2P)

เพียร์ทูเพียร์ คือการประมวลผลแบบหนึ่งที่คอมพิวเตอร์แต่ละเครื่องในระบบเครือข่ายมีหน้าที่และความรับผิดชอบเท่ากัน ซึ่งต่างจากสถาปัตยกรรมแบบไคลเอนท์เซิร์ฟเวอร์(client/server) ที่จะมีเครื่องคอมพิวเตอร์เครื่องหนึ่งเรียกว่าเซิร์ฟเวอร์ ทำหน้าที่รับการร้องขอจากเครื่องคอมพิวเตอร์เครื่องอื่นที่เรียกว่าไคลเอนท์เพื่อให้บริการต่างๆ แต่ในเพียร์ทูเพียร์นั้นทุกเครื่องจะให้บริการกันเองจึงมีรูปแบบของเครือข่ายที่ง่ายกว่า ซึ่งเพียร์ทูเพียร์มีข้อจำกัดคือไม่เหมาะกับงานที่หนักมากๆ

2.2.3. คลัสเตอร์(Cluster)

คลัสเตอร์เกิดจากแนวความคิดที่จะนำกลุ่มของคอมพิวเตอร์หลายๆเครื่องที่เชื่อมต่อกันด้วยเครือข่ายมาช่วยกันทำงานที่มีขนาดใหญ่งานหนึ่ง โดยคลัสเตอร์ประกอบด้วยเครื่องคอมพิวเตอร์หลายเครื่องซึ่งแต่ละเครื่องมีซีพียูและหน่วยความจำของตัวเองเชื่อมต่อกันโดยเครือข่ายในการสื่อสาร (LAN)

เพื่อใช้ทรัพยากรร่วมกันในการทำงานที่มีขนาดใหญ่โดยเครื่องแต่ละเครื่องจะทำการประมวลผลแยกจากกันอย่างเป็นอิสระและหน่วยความจำจะถูกแบ่งให้กับตัวประมวลผลแต่ละตัว และเครื่องแต่ละเครื่องจะติดต่อกันโดยการรับและส่งแมสเซส คลัสเตอร์ทำให้เราสามารถใช้ประโยชน์จากเครื่องที่มีอยู่ได้อย่างเต็มที่ และทำให้เราสามารถทำการประมวลผลแบบขนานโดยไม่ต้องใช้ซุปเปอร์คอมพิวเตอร์ที่มีราคาแพง และเนื่องจากในองค์กรส่วนมากมักจะมี LAN ที่มีความเร็วสูงเชื่อมต่อกับเครื่องจำนวนมากจึงอาจจะทำให้เกิดกำลังในการประมวลผลมากกว่าซุปเปอร์คอมพิวเตอร์เลยก็ได้

โครงสร้างของระบบคลัสเตอร์นั้นแบ่งออกเป็น 3 ส่วนหลักๆ คือ ฮาร์ดแวร์,ซอฟแวร์ และมิดเดิ้ลแวร์

2.2.3.1. Beowulf Cluster

Beowulf cluster คือกลุ่มของคอมพิวเตอร์ที่เชื่อมต่อกันผ่านเครือข่ายและทำการประมวลผลแบบขนานโดยBeowulf เป็นคอมพิวเตอร์แบบขนานที่มีประสิทธิภาพมากซึ่งไม่ได้เกิดจากการฮาร์ดแวร์ที่มีประสิทธิภาพสูง จะทำงานบนระบบปฏิบัติการอย่างเช่น Linux หรือ FreeBSD เชื่อมต่อโดยเครือข่ายที่มีความเร็วสูง โดยจะประกอบด้วยคลัสเตอร์ของworkstation เพื่อทำงานในการประมวลผลที่มีขนาดใหญ่

ในการใช้งาน Beowulf นั้นแอพพลิเคชั่นที่มีอยู่จะต้องถูกแตกออกเป็นงานที่ทำงานแบบขนานที่ติดต่อกันโดยใช้ MPI หรือ PVM โดยระบบจะประกอบด้วยฮาร์ดแวร์ที่เป็นเซิร์ฟเวอร์ 1 เครื่อง และไคลเอนท์อย่างน้อย 1 เครื่อง เชื่อมต่อกันด้วยระบบเครือข่ายภายใน เช่น Ethernet

รูปที่ 2-1 ตัวอย่างการสร้าง Beowulf Cluster **ภายในบ้าน**

2.2.3.2. ข้อเปรียบเทียบระหว่างกริดกับคลัสเตอร์

หัวข้อ	คลัสเตอร์	กริด
จำนวนเครื่องคอมพิวเตอร์	มากกว่า 1 เครื่อง	มากกว่า 1 เครื่อง
การทำคอมพิวเตอร์เสมือน	ทำ	ทำ
การประมวลผล	ขนาน	ขนาน
การใช้มิดเดิ้ลแวร์	ใช้	ใช้
เครือข่าย	LAN (จำกัด)	internet
จำนวนadmin	จำกัดเพียง1admin	ไม่จำกัด
จุดมุ่งหมาย	คำนวณงานขนาดใหญ่ และสมรรถนะในการคำนวณ	กระจายงานไปยังเครื่องอื่นๆ และวิธีการแชร์ทรัพยากร
ชนิดของโปรเซสเซอร์	ชนิดเดียวกัน	แตกต่างกันได้
ชนิดของระบบปฏิบัติการ	เหมือนกัน	แตกต่างกันได้
ลักษณะของทรัพยากร	static	dynamic

ตารางที่ 2-1 ข้อเปรียบเทียบระหว่างคลัสเตอร์และกริด

2.2.4. มาตรฐานที่ใช้ในการส่งแมสเซส

เมื่อการประมวลผลแบบขนานได้รับความนิยมมากและความแตกต่างทาง สถาปัตยกรรมของเครื่องคอมพิวเตอร์จึงทำให้เกิดมาตรฐานขึ้นเพื่อให้การส่งแมสเซสมีประสิทธิภาพมากขึ้น คือ

2.2.4.1. PVM (Parallel Virtual Machine)

PVM เป็นซอฟแวร์แพคเกจ ที่ทำให้กลุ่มของเครื่องที่ต่างกันซึ่งเชื่อมต่อกันด้วยเครือข่ายที่ประกอบด้วยสถาปัตยกรรมที่ต่างกัน PVM มีจุดประสงค์เพื่อให้สภาพแวดล้อมที่ต่างกันสามารถติดต่อกันได้ สถาปัตยกรรมแบบหนึ่งสามารถถูกคัดลอกไปยังเครื่องที่มีสถาปัตยกรรมอีกแบบแล้วทำการคอมไพล์(compile) และเอกซีคิวท์(execute) ได้โดยไม่ต้องทำการดัดแปลงเพื่อให้เครื่องหลายๆเครื่องติดต่อกันเหมือนเป็นคอมพิวเตอร์แบบขนาน PVM ได้กำหนดระบบปฏิบัติการแบบกระจาย โดยจะมีเดมอน (daemon) โพรเซสทำงานบนเครื่องคอมพิวเตอร์ทั้งหมดเพื่อให้เกิด virtual machine และ PVM นี้ master จะถูกสร้างใน local machine และจะสร้างเดมอน (daemon) ในเครื่องอื่นๆโดยอัตโนมัติ แต่ใน MPI นั้น master กับ slave จะถูกสร้างขึ้นพร้อมกัน

2.2.4.2. MPI (Message Passing Interface)

MPI ถูกสร้างขึ้นเพื่อพัฒนามาตรฐานในการเขียนโปรแกรมส่งแมสเซสโดย MPI Forum MPI มีคุณสมบัติดังนี้

- MPI เป็น library เป็นมาตรฐานในการส่งแมสเซส

- ไม่ต้องทำการดัดแปลงซอสโค้ด(source code) เมื่อจะนำแอพพลิเคชั่นไปใช้กับแพลตฟอร์ม (platform) อื่นที่สนับสนุน MPI
- การอิมพลีเมนต์(implementation)ของผู้ผลิต(vendor)จะสามารถใช้ลักษณะเด่นของฮาร์ดแวร์ที่มีอยู่เพื่อให้เกิดประสิทธิภาพได้อย่างเต็มที่
- มีจำนวนฟังก์ชั่นมากกว่า 115 รูทีน(routine) และจะเพิ่มขึ้นได้โดยขึ้นอยู่กับ ประสบการณ์จากระบบการส่งข้อมูลต่างๆที่ใช้งาน MPI

2.2.5. แนวความคิดในการจัดการทรัพยากร

เมื่อมีการใช้ทรัพยากรร่วมกันจึงเกิดแนวความคิดในการจัดการในการใช้งานทรัพยากรได้แก่

2.2.5.1. Mutex (mutual exclusion object)

Mutex เป็นโปรแกรมที่ยอมให้ thread หลายๆ thread แชร์ทรัพยากรเดียวกัน thread ที่ต้องการทรัพยากรต้อง lock mutex จาก thread อื่นขณะที่มันกำลังใช้ทรัพยากร และ mutex จะ unlock เมื่อมันไม่ได้ใช้งานเป็นเวลานานหรือ thread สิ้นสุดลง ความแตกต่างระหว่าง mutex กับ semaphore คือ mutex ถูกเป็นเจ้าของโดย thread ซึ่ง lock มัน และมีเพียงโพรเซสที่ทำการ lock ที่จะสามารถ unlock ได้เท่านั้นขณะที่ semaphore สามารถถูกเปลี่ยนสถานะโดย thread หรือโพรเซสอื่นได้

2.2.5.2. Semaphore

Semaphore เป็นวิธีที่จะควบคุมการสื่อสารระหว่าง 2 โพรเซสที่ใช้ทรัพยากรร่วมกันเพื่อให้มีเพียงโพรเซสเดียวที่สามารถเข้าถึงทรัพยากรได้ในเวลาหนึ่งๆการจองทรัพยากรทำได้โดยกำหนดค่าของ semaphore ค่านี้จะทำหน้าที่กำหนดจำนวนโพรเซสที่สามารถใช้ทรัพยากรได้ในช่วงเวลาหนึ่งๆ โดยโพรเซสจะต้องได้รับค่า semaphore ก่อนจึงจะใช้ทรัพยากรได้โดยจะทำการลดค่าของ semaphore และมันจะปล่อย semaphore เมื่อใช้ทรัพยากรเสร็จแล้วโดยจะทำการเพิ่มค่า semaphore และเพื่อที่จะได้รับทรัพยากรที่ถูกแชร์ โพรเซสจะต้องทำการทดสอบ semaphore ที่ควบคุมทรัพยากรโดย

- ถ้าค่าของ semaphore เป็นบวกแล้วโพรเซสสามารถใช้ทรัพยากรได้ และโพรเซสจะลด semaphore ลงหนึ่งเพื่อบ่งบอกว่าใช้ทรัพยากรไปแล้วหนึ่งหน่วย
- ถ้าค่าของทรัพยากรเป็นศูนย์แล้วโพรเซสจะทำการ sleep จนกว่าค่า semaphore จะเพิ่มมากกว่าศูนย์ และเมื่อโพรเซส wake up จะกลับไปทำการตรวจสอบค่า semaphore ใหม่เมื่อโพรเซสใช้ทรัพยากรที่ถูกแชร์เสร็จแล้วค่าของ semaphore จะเพิ่มขึ้น 1 และโพรเซสที่ sleep อยู่จะถูก awake เพื่อให้ทำงานต่อได้

2.2.6. เทคโนโลยีอื่นๆในปัจจุบัน

ปัจจุบันเทคโนโลยีของซีพียูได้มีการพัฒนาไปมากจนเกิดเทคโนโลยีใหม่ๆ ได้แก่

2.2.6.1. Dual Core

สถาปัตยกรรมแบบ Dual Core คือการติดตั้ง 2 Core บนซีพียูตัวเดียวกัน เพื่อเพิ่มประสิทธิภาพการทำงานคล้ายกับ Dual CPU ซึ่งมี 2 ซีพียูในเครื่องๆเดียว แต่ Dual Core จะมีความเร็วมากกว่าเพราะสามารถส่งผลลัพธ์จาก core หนึ่งไปยังอีก core หนึ่งโดยไม่ต้องใช้system bus

2.2.6.2. Hyper-threading

เทคโนโลยี hyper-threading นี้จะใช้ hyperthread CPU ซึ่งสามารถจัดการกับชุดคำสั่งในหลายๆ thread ได้โดยไม่ต้องอาศัย system bus โดยจะมีหน่วยที่ทำหน้าที่จัดเรียงคำสั่งก่อนการประมวลผลที่ดี ซึ่งทำให้สามารถใช้ execution resource ชุดเดียวใน การประมวลผลชุดคำสั่งในหลายๆ thread ไปพร้อมๆกันได้ โดยแทรกคำสั่งของแต่ละ thread ลงใน execution resource อย่างเหมาะสมโดยที่ชุดคำสั่งแต่ละชุดยังเป็นอิสระจากกัน

2.3. ประเภทของกริด

2.3.1. Computational Grid

Computational Grid จะเน้นไปที่กำลังในการประมวลผล เครื่องที่ใช้มักจะเป็นเครื่องเซิร์ฟเวอร์ที่มีประสิทธิภาพสูง

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

องค์กรส่วนใหญ่ที่ใช้ Computational Grid เกิดจากมีการริเริ่มทางธุรกิจในการขยายความสามารถและเพิ่มการใช้ทรัพยากรที่มีอยู่ของคอมพิวเตอร์โดยการรวมและแชร์ทรัพยากร ในธุรกิจอาจจะต้องการความจุของคอมพิวเตอร์มากกว่าจำนวนที่มีอยู่ จึงมีธุรกิจที่สนใจในการดัดแปลงแอพพลิเคชั่นเพื่อใช้การประมวลผลแบบขนาน

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

2.3.2. Scavenging Grid

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

2.3.3. Data Grid

Data Grid ดูแลและควบคุมการเข้าถึงข้อมูลในหลายองค์กร โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ตำแหน่งที่อยู่ของข้อมูลตลอดระยะเวลาที่ใช้ข้อมูล ตัวอย่างเช่น มหาวิทยาลัยสองแห่งทำการวิจัยวิทยาศาสตร์

ชีวภาพโดยใช้ข้อมูลเดียวกัน Data Grid จะช่วยในการแชร์ข้อมูล, จัดการข้อมูล, รักษาความปลอดภัย อย่างเช่น กำหนดว่าผู้ใช้แต่ละคนสามารถเข้าไปใช้ข้อมูลส่วนไหนได้บ้าง, และกำหนดการเข้าถึงข้อมูลที่น่าเชื่อถือและรวดเร็วธุรกิจที่ใช้ Data Grid มักจะเกิดจากการริเริ่มต้นทางสารสนเทศเพื่อขยายความสามารถของเหมืองแร่ข้อมูล และเพิ่มประโยชน์ในการลงทุนโครงสร้างพื้นฐานในการเก็บข้อมูลที่มีอยู่และลดความซับซ้อนในการจัดการข้อมูล

2.4. การทำงานของกริด

2.4.1. การใช้ประโยชน์จากทรัพยากรที่ว่าง

ในองค์กรส่วนมากมักจะมีทรัพยากรส่วนหนึ่งที่ถูกใช้งานอย่างเต็มที่ ขณะที่ทรัพยากรอีกส่วนหนึ่งไม่ได้ใช้งานกริดจึงนำทรัพยากรที่ไม่ได้ใช้งานเหล่านั้นมาใช้งานเพื่อให้การทำงานมีประสิทธิภาพเพิ่มขึ้นและยังเป็นการใช้ทรัพยากรที่มีอยู่ให้เกิดประโยชน์สูงสุด

2.4.2. การทำงานแบบขนาน

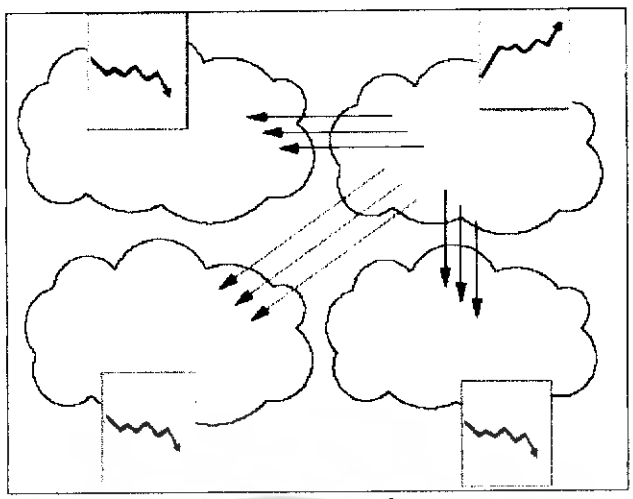
กริดจะทำการแตกงานออกเป็นหลายๆส่วนโดยแต่ละส่วนจะทำงานแยกจากกันอย่างเป็นอิสระและแยกกันไปทำงานในเครื่องที่อยู่ในที่ต่างๆกันภายในกริด แต่ก็มีงานบางประเภทที่ไม่สามารถแตกออกจากกันอย่างเป็นอิสระได้ อย่างเช่น งานที่ต้องรอผลลัพธ์จากอีกส่วนหนึ่งก่อนจึงจะสามารถทำงานต่อได้

2.4.3. การสร้างองค์กรเสมือน(Virtual Organization)

กริดได้สร้างมาตรฐานที่ทำให้ระบบที่ต่างกันสามารถทำงานด้วยกันได้เพื่อสร้างภาพของระบบการประมวลผลเสมือนขนาดใหญ่ที่มีทรัพยากรที่หลากหลาย โดยผู้ใช้กริดจะถูกจัดเข้าไปอยู่ในองค์กรเสมือนจำนวนหนึ่งซึ่งแต่ละองค์กรมีนโยบายที่ต่างกัน องค์กรเสมือนเหล่านี้สามารถแชร์ทรัพยากรร่วมกันเพื่อเป็นกริดที่มีขนาดใหญ่ขึ้นผู้มีส่วนร่วมและผู้ใช้กริดอาจจะเป็นสมาชิกขององค์กรจริงและองค์กรเสมือนที่ต่างกันได้หลายองค์กร กริดจะช่วยในการบังคับใช้กฎความปลอดภัยระหว่างองค์กรและสร้างกฎที่ช่วยจัดลำดับการใช้งานของทรัพยากรและผู้ใช้

2.4.4. การสร้างสมดุลให้กับทรัพยากร

กริดจะสร้างความสมดุลให้กับทรัพยากรโดยการจัดลำดับงานของกริดให้กับเครื่องที่ว่างโดยในช่วงที่มีงานจำนวนมาก งานบางส่วนจะถูกส่งไปยังเครื่องที่ว่างเพื่อช่วยในการประมวลผลและถ้ากริดถูกใช้งานอย่างเต็มที่แล้ว งานที่มีลำดับความสำคัญต่ำสุดจะถูกพักไว้ชั่วคราวหรือยกเลิกไปก่อนเพื่อจะได้ทำงานที่มีลำดับความสำคัญสูงกว่าก่อน โดยการสร้างความสมดุลให้กับทรัพยากรจะช่วยลดความคับคั่งหรือระยะเวลาในการสื่อสาร



รูปที่ 2-2 งานถูกย้ายไปยังส่วนที่มีงานน้อยกว่าเพื่อสร้างความสมดุลให้กับทรัพยากร

2.4.5. การจัดการข้อมูลภายในกริด

เมื่อแอพพลิเคชั่นกริดถูกแตกออกเป็นงานย่อยแล้ว ข้อมูลอินพุทเป็นชุดข้อมูลที่มีขนาดจำกัดจึงทำให้เกิดการแชร์ข้อมูล โดยจะกำหนดที่ตั้งของข้อมูลที่แชร์ให้ใกล้กับงานที่ต้องการข้อมูลนั้นๆถ้าข้อมูล ถูกใช้มากกว่าหนึ่งครั้งมันจะถูกทำสำเนาเก็บไว้หลายที่ในจำนวนที่มีพื้นที่เพียงพอที่จะเก็บได้

ข้อมูลที่แชร์อาจจะมีค่าคงที่หรือเปลี่ยนแปลง ตัวอย่างเช่นฐานข้อมูลอาจจะประกอบด้วยลำดับยีนที่เพิ่งพบไม่นานมานี้และกำลังเพิ่มขึ้นอยู่เสมอๆ อย่างไรก็ตามแอพพลิเคชั่นที่ใช้ข้อมูลนี้อาจจะยังไม่ต้องการข้อมูลลำดับยีนล่าสุดนี้โดยทันที ดังนั้นทำให้เป็นการง่ายขึ้นและมีประสิทธิภาพมากขึ้นในการแชร์ฐานข้อมูลในการปรับปรุงข้อมูลจะถูกรวมกลุ่มไว้และทำการประมวลผลในเวลาที่ไม่ได้ใช้งานสูงสุดนอกจากนั้นถ้ามีสำเนาข้อมูลมากกว่า 1 ชุดแล้วสำเนาไม่ต้องถูกปรับปรุงพร้อมๆกันทำให้ประสิทธิภาพดีขึ้นเพราะแอพพลิเคชั่นทั้งหมดที่ใช้ข้อมูลจะไม่ต้องถูกหยุดขณะที่กำลังปรับปรุงข้อมูล มีเพียงแค่การเข้าใช้สำเนาบางอันเท่านั้นที่ต้องถูกหยุดหรือพักไว้ชั่วคราว

เมื่อไฟล์หรือฐานข้อมูลถูกปรับปรุง งานจะไม่สามารถอ่านข้อมูลของไฟล์ได้ในนั้นโดยงานอื่นจะถูก lock หรือการทำไปพร้อมๆกันเพื่อป้องกันมันโดยอัตโนมัติเพื่อไม่ให้แอพพลิเคชั่นอ่านข้อมูลซึ่งจะทำให้ได้รับทั้งข้อมูลเก่าและใหม่ ในสถานการณ์ที่ข้อมูลถูกแชร์การปรับปรุงต้องไปถูกเลื่อนออกไป ตัวอย่างเช่นถ้างานย่อยกำลังประมวลผลการดำเนินงานทางการเงิน จะต้องถูกปรับปรุงโดยทันทีในฐานข้อมูลหลัก นอกจากนั้นถ้ามีสำเนาของฐานข้อมูลที่ใด จะต้องถูกปรับปรุงด้วยข้อมูลใหม่โดยทันทีจะต้องทำให้ข้อมูลจำนวนมากตรงกันในการสื่อสารระหว่างงานกับฐานข้อมูล ซึ่งจะทำให้เกิดปัญหา bottleneck ซึ่งส่งผลต่อประสิทธิภาพของกริดทั้งหมด จึงต้องพิจารณาถึงวิธีที่จะแบ่งกิจกรรมของฐานข้อมูลเพื่อให้มีการแทรกแซงน้อยในส่วนต่างๆและมีการจัดการเพื่อทำให้ข้อมูลตรงกัน

2.4.6. การจัดการเรื่องความปลอดภัยภายในกริด

เนื่องจากมีการติดต่อรวบรวมและกระจายทรัพยากรออกไป ทำให้เป้าหมายที่สำคัญอย่างหนึ่งที่กริดต้องมีคือกระบวนการจัดการด้านความปลอดภัยในระบบซึ่งบริการเหล่านั้น ซึ่งจะกล่าว ในรายละเอียดในบทที่ 4

2.5. ทรัพยากรที่ใช้ในกริด

- หน่วยประมวลผล
- หน่วยเก็บข้อมูล

  เครื่องแต่ละเครื่องในกริดมักจะกำหนดปริมาณที่สามารถใช้เก็บข้อมูลในกริดได้โดยไฟล์หรือฐานข้อมูลของเครื่องๆหนึ่งสามารถกระจายไปยังอุปกรณ์เก็บข้อมูลหรือเครื่องหลายเครื่องได้และผู้ใช้สามารถอ้างถึงข้อมูลในกริดได้อย่างง่ายโดยไม่ต้องรู้ตำแหน่งที่แท้จริงของข้อมูลนั้น
- แบนด์วิธในการสื่อสารข้อมูล

  แบนด์วิธที่ใช้ในการสื่อสารเป็นทรัพยากรที่สำคัญซึ่งมีอยู่จำกัดในการใช้งานกริด เส้นทางในการสื่อสารที่มีจำนวนมากจะช่วยควบคุมความผิดพลาดของเครือข่ายและความคับคั่งของข้อมูล และจะแก้ไขปัญหา bottleneck ในการสื่อสาร
- ซอฟท์แวร์

  ซอฟท์แวร์บางชนิดมีลิขสิทธิ์และมีราคาแพงมากกว่าที่จะติดตั้งบนเครื่องทุกเครื่องในกริดโดยในกริดนั้นงานที่ต้องใช้ซอฟท์แวร์นี้ในการทำงานจะถูกส่งไปยังเครื่องที่มีซอฟท์แวร์นี้ติดตั้งอยู่แล้ว เพื่อลดค่าใช้จ่ายขององค์กรและนอกจากนี้ลิขสิทธิ์ซอฟท์แวร์บางตัวยอมให้ติดตั้งได้บนทุกเครื่อง ของกริดแต่จะจำกัดจำนวนเครื่องที่สามารถใช้งานพร้อมกันได้ในเวลาขณะหนึ่ง
- อุปกรณ์ต่างๆ เช่น พรินเตอร์
- และอื่นๆ

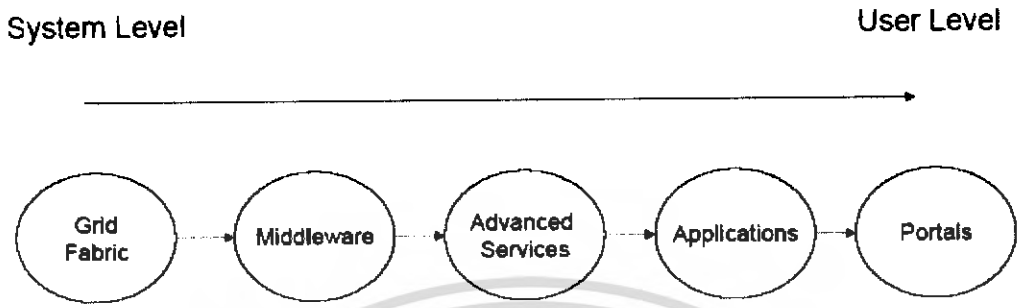
2.6. สถาปัตยกรรมของกริด

กริดแสดงให้เห็นลักษณะที่แตกต่างของทางสถาปัตยกรรม ที่แสดงให้เห็นทิศทางที่ควรจะเป็นไป ของกริด และพบว่าสถาปัตยกรรมของกริดเป็นลักษณะของ Multi-faceted แบ่งได้เป็น 3 ประเภทคือ

2.6.1. สถาปัตยกรรมกริดแบบลำดับชั้น(N-Tier Grid Architecture)

สถาปัตยกรรมกริดแบบลำดับชั้น เป็นลักษณะของแบบจำลองเพื่อให้การพัฒนากริดได้รับความยืดหยุ่นและสามารถ นำแอพพลิเคชั่นที่เคยสร้างไว้กลับมาพัฒนา และใช้งานซ้ำได้การแบ่งแยกกริดออกเป็นชั้นๆ ทำให้ผู้พัฒนาสามารถทำการเปลี่ยนแปลงและเพิ่มเลเยอร์ที่ต้องการได้ จึงทำให้นักพัฒนา

เลือกเพิ่มเติมหรือแก้ไขได้อย่างจำเพาะเจาะจงแทนที่จะต้องทำการสร้างแอพพลิเคชั่นขึ้นมาใหม่ ซึ่งเปรียบได้กับชั้นที่ 7 ของ OSI model จึงเป็นที่นิยมใช้กันโดยทั่วไป

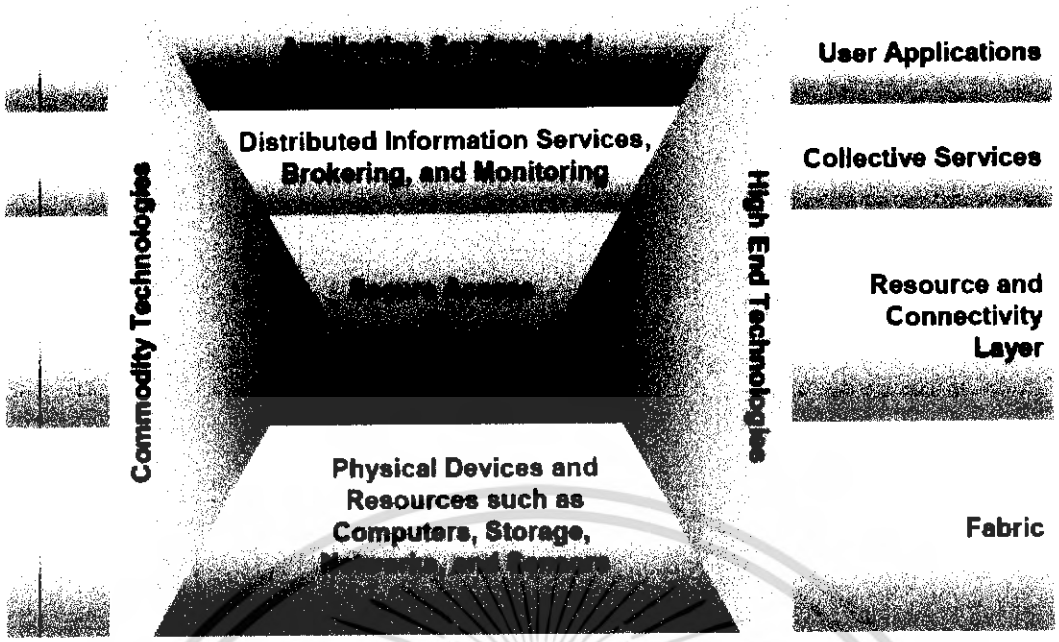


รูปที่ 2-3 รูปสถาปัตยกรรมกริดแบบลำดับชั้น

2.6.2. Role-Based Grid Architecture (สถาปัตยกรรมกริดที่อ้างอิงกับหน้าที่)

การเข้าถึงกลุ่มของทรัพยากรทางกายภาพได้อย่างปลอดภัย จะต้องใช้สถาปัตยกรรมแบบที่อ้างอิงหน้าที่ ซึ่งสถาปัตยกรรมแบบนี้ง่ายที่จะบ่งบอกส่วนประกอบต่างๆของระบบที่ทำการติดต่อกับภายนอกซึ่งเปรียบเสมือนเป็นโปรโตคอลที่แยกออกเป็นประเภทๆตามหน้าที่โดยร่วมใช้ข้อมูลชุดเดียวกัน

สถาปัตยกรรมแบบนี้แบ่งออกได้เป็น 5 ชั้น คือ fabric, connectivity, resource, collective, and application layer และใช้มาตรฐานเหล่านี้เพื่อช่วยให้การแลกเปลี่ยนข้อมูลเป็นไปด้วยความปลอดภัย และเสมือนเป็นทรัพยากรจากแหล่งเดียวกันทรัพยากรเหล่านี้ถูกควบคุมและดูแลโดย collective services ซึ่งทำให้มองเสมือนทรัพยากรเหล่านั้นคือทรัพยากรจากแหล่งเดียวกันเพื่อให้ง่ายต่อการออกแบบแอพพลิเคชั่นและง่ายสำหรับการเรียกใช้ของผู้ใช้งาน



รูปภาพที่ 2-4 รูปสถาปัตยกรรมกริดที่อ้างอิงกับหน้าที่

ในสถาปัตยกรรมแบบที่ขึ้นกับหน้าที่แบ่งเป็นเลเยอร์ต่างๆดังนี้

- Fabric layer

  ประกอบไปด้วยโปรโตคอลต่างๆที่ทำการติดต่อกับ แอพพลิเคชั่นและเครื่องมือที่ทำให้เกิดการพัฒนาการบริการต่างๆ รวมไปถึงควบคุมส่วนต่างๆที่ทำการเข้าถึงทรัพยากร เช่น เครื่องคอมพิวเตอร์ , แหล่งเก็บทรัพยากร , ระบบเครือข่าย และ เซ็นเซอร์ เป็นต้น โดยควรจะมีการกำหนดกลไก enquiry ซึ่งจะหาโครงสร้างและสถานะ รวมถึงกลไกการจัดการทรัพยากรซึ่งควบคุมคุณภาพของการให้บริการ(quality of service)

- Connectivity layer

  เป็นส่วนที่รับผิดชอบหน้าที่ในการติดต่อ และ รับรองการเชื่อมต่อของเครือข่าย เพื่อทำให้เกิดความปลอดภัยในระหว่างที่ทำการติดต่อกับทรัพยากรของกริด เป็นโปรโตคอลและบริการที่มีหน้าที่แลก เปลี่ยนข้อความที่มีความปลอดภัย , การรับรองสิทธิ ,และการให้อำนาจในการติดต่อ

- Resource layer

  ประกอบไปด้วยโปรโตคอลที่ทำหน้าที่ดูแลและสร้างความปลอดภัยให้กับทรัพยากร โปรโตคอล Resource layer แบ่งออกเป็น

  - Information protocols (โปรโตคอลข้อมูล)

    ถูกใช้เพื่อรับข้อมูลเกี่ยวกับโครงสร้างและสถานะของทรัพยากร เช่นการ configuration, load ปัจจุบัน และกฎการใช้งาน

- Management protocols (โปรโตคอลในการจัดการ)

  ถูกใช้เพื่อควบคุมการเข้าถึงทรัพยากรที่ถูกแชร์ เช่น การร้องขอทรัพยากร (รวมถึงการจองล่วงหน้าและการควบคุมคุณภาพของการให้บริการ) และการทำ operation อย่างเช่น การสร้างโพรเซสหรือการเข้าถึงข้อมูล

- Collective layer

  รับผิดชอบในส่วนของการทำงานร่วมกันของทรัพยากรจากหลายแหล่ง และบ่งบอกว่าอยู่ส่วนไหนในองค์กรเสมือน และสามารถกำหนดพฤติกรรมในการแชร์ได้

- Application layer

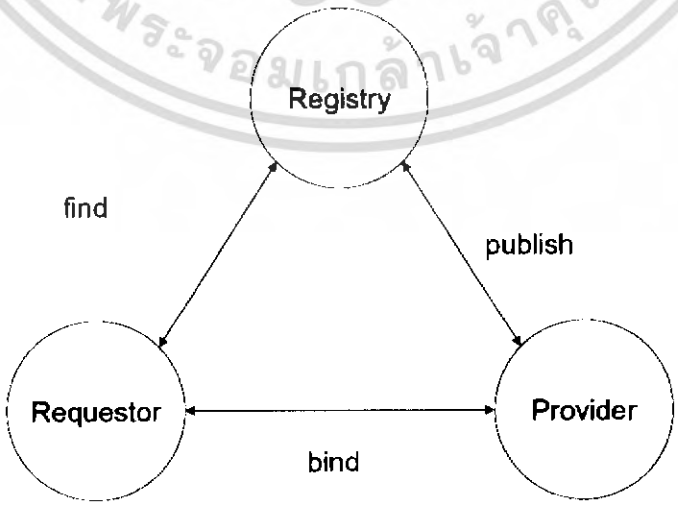
  ทำหน้าที่ติดต่อกับผู้ใช้โดยตรงโดยผ่านแอพพลิเคชั่นซึ่งจะติดต่อผ่านทางองค์กรเสมือนอีกทีหนึ่ง

2.6.3. Service-Based Grid Architecture (สถาปัตยกรรมกริดที่อ้างอิงกับเซอร์วิส)

ปัจจุบันเราพบว่าแนวทางของเทคโนโลยีจะเน้นไปทางแนวคิดที่แต่ละส่วนของบริการใดๆเป็นอิสระต่อกัน และภาพรวมของกริดพบว่าลักษณะซอฟท์แวร์ที่เราต้องการคือไม่ขึ้นอยู่กับแพลตฟอร์มใดๆ บริการที่เราต้องการจะสามารถเรียกถึงได้ง่ายซึ่งกระบวนการนี้เรียกว่า การค้นหารีซอร์ส (resource discovery)

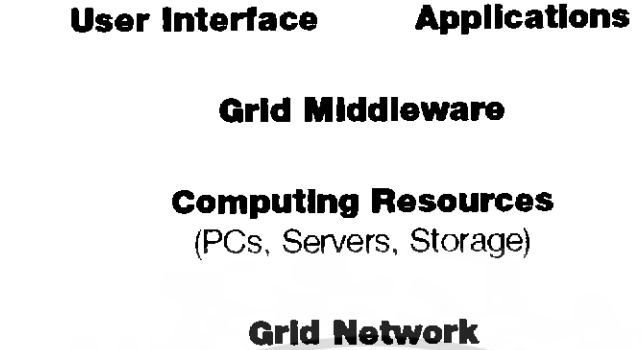
ประโยชน์ของสถาปัตยกรรมกริดที่ขึ้นกับเซอร์วิส เปรียบได้กับกลุ่มของคอมพิวเตอร์ที่ทำงานตามรูปแบบตารางเวลา เริ่มจาก หากลุ่มรีซอร์สที่สามารถใช้งานได้ ต่อมาก็เลือกรีซอร์สที่ใช้คำนวณจากกลุ่มที่ได้ในขั้นแรกเพื่อจัดตารางงาน โดยหลักการเลือกรีซอร์สนั้น อาจเลือกจากอัตราการโหลดของแต่ละรีซอร์ส หรือ จากค่าต่างๆ ตามแต่จะพิจารณาว่าสมควร

โดยสรุปแล้วโครงสร้างนี้สามารถพัฒนาเป็นการทำงานที่เชื่อมต่อกันระหว่างเซอร์วิสได้ เพราะรูปแบบที่ขึ้นกับเซอร์วิสนี้ เป็นเซอร์วิสที่ไม่ได้เกิดขึ้นในเวลาเดียวกันตามรูปแบบ asynchronous service นั่นเอง



รูปที่ 2-5 รูปสถาปัตยกรรมกริดที่อ้างอิงกับเซอร์วิส

2.7. โครงสร้างของระบบการประมวลผลแบบกริด



รูปที่ 2-6 โครงสร้างของระบบการประมวลผลแบบกริด

โครงสร้างของกริดนั้น จะประกอบไปด้วย

2.7.1. User Interface & Application

เป็นแอพพลิเคชั่นที่ติดต่อกับผู้ใช้งานระบบกริด เช่น ส่วนที่ใช้จัดการควบคุมดูแลกริด ส่วนที่ใช้จัดหาเซอร์วิส เป็นต้น

2.7.2. Grid Middleware

เป็นเครื่องมือที่ช่วยในการสร้างระบบกริด เปรียบเสมือนส่วนที่เชื่อมต่อระหว่างผู้ใช้งานกับทรัพยากรที่ใช้งานในกริด มีหลายเครื่องมือที่ใช้เป็นมิดเดิ้ลแวร์ เช่น Globus Toolkit, Rocks Cluster & Roll Grid เป็นต้น

2.7.3. Computing Resources

เป็นทรัพยากรที่ใช้ในการคำนวณ ประกอบไปด้วย คอมพิวเตอร์ แหล่งเก็บข้อมูล แหล่งประมวลผลต่างๆ เป็นต้น

2.7.4. Grid Network

ระบบเครือข่ายที่ใช้ในกริด ซึ่งรวมถึงระบบเครือข่ายภายใน (Local Area Network) และรวมถึงระบบเครือข่ายภายนอก (Wide Area Network)

ส่วนที่สำคัญที่สุดที่จะต้องกล่าวถึงเป็นพิเศษก็คือ มิดเดิ้ลแวร์ เนื่องจากเป็นเสมือนส่วนที่ทำให้ระบบกริด สามารถใช้งานได้จริง ซึ่งมิดเดิ้ลแวร์ที่นิยมใช้ คือ Globus Toolkit ซึ่งมีข้อดีก็คือ ได้นำเทคโนโลยีเว็บเซอร์วิสมาผนวกรวมเข้ากับมาตรฐานที่ใช้กำหนดระบบกริด (Open Grid Service Architecture หรือ OGSA) และยังเป็นเครื่องมือที่ผู้คิดค้นกริดที่รวมตัวเป็นกลุ่มๆหนึ่ง ที่เรียกว่า Global Grid Forum หรือ GFF พัฒนาขึ้นมาโดยตรงด้วย

ข้อดีของการนำกริดมาอ้างอิงบนพื้นฐานของ เว็บเซอร์วิส คือจะทำให้ได้เฟรมเวิร์ค ที่สามารถขยายได้เพื่อการเข้าถึง ทรัพยากรของกริด โดยใช้มาตรฐานต่างๆที่มีอยู่แล้ว เช่น SOAP, XML และ WS-Security, TCP/IP และ มาตรฐาน IETF และอื่นๆ ทำให้สามารถติดต่อสื่อสารแบบ interoperable ระหว่างระบบที่แตกต่างกันได้ในอินเตอร์เน็ตและเครือข่ายที่ใช้

2.8. ลักษณะของแอพพลิเคชั่นที่เหมาะแก่การใช้งานกริด

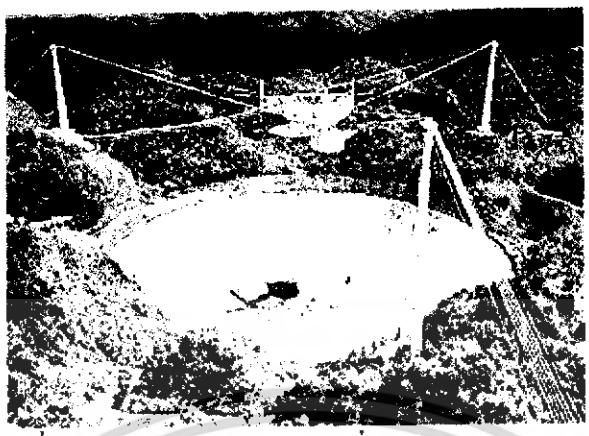
ถึงแม้ว่าการใช้งานกริดจะเกิดประโยชน์อย่างมาก แต่ก็มีแอพพลิเคชั่นบางชนิดที่ไม่เหมาะแก่การใช้งานกริดเนื่องจากมันใช้ทรัพยากรในการประมวลผลจำนวนไม่มาก ซึ่งถ้าใช้กริดในการทำงานอาจจะเป็นการลดประสิทธิภาพในการทำงานของมันเนื่องจากต้องเสียเวลาในการส่งข้อมูลผ่านทางเครือข่ายและควรพิจารณาว่างานนั้นเป็นการคำนวณที่สามารถทำงานในแบบขนานได้หรือไม่ อย่างเช่น ในการประมวลผลที่เกี่ยวกับการคำนวณบางอย่างอาจจะไม่ต้องทำตามลำดับก่อนหลัง จึงสามารถทำงานแบบขนานได้แต่ก็มีงานบางประเภทที่ต้องรอผลลัพธ์จากอีกส่วนหนึ่งก่อนจึงจะสามารถทำการประมวลผลต่อได้ซึ่งไม่เหมาะกับการใช้งานกริด

2.9. ตัวอย่างของงานที่ใช้การประมวลผลแบบกริด

2.9.1. SETI@Home หรือ The Search for Extraterrestrial Intelligence

SETI@Home เป็นโครงการของ University of California at Berkeley โดยที่ UC. Berkeley มีโครงการ SETI ที่ค้นหาอารยธรรมนอกโลกมานานแล้วโดยใช้กล้องโทรทรรศน์วิทยุเฝ้าฟังสัญญาณเพื่อนำมาวิเคราะห์สัญญาณซึ่งจำเป็นต้องใช้เครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงมากๆระดับซุปเปอร์คอมพิวเตอร์ แต่เนื่องจากซุปเปอร์คอมพิวเตอร์มีราคาแพงมาก ทีม SETI ของ UC. Berkeley จึงเกิดแนวความคิดที่จะนำเครื่อง PC ทั่วไปซึ่งมักจะไม่ได้ใช้ CPU time เต็ม 100% มาร่วมกันประมวลผลข้อมูลจึงเกิดโครงการ SETI@Home ขึ้น โดย SETI@Home ใช้หลักการของการประมวลผลแบบกระจาย (distributed computing) แบบไคลแอนท์เซิร์ฟเวอร์ (client/server) โดยจะมีโปรแกรมที่เป็นไคลเอนท์ ของ SETI@Home ติดตั้งไว้ที่เครื่อง PC และตัวโปรแกรมนี้จะติดต่อกับเซิร์ฟเวอร์ ที่ UC. Berkeley เพื่อรับงานไปประมวลผลและส่งผลลัพธ์กลับมา SETI@Home ให้อาสาสมัครจากทั่วโลกที่มีคอมพิวเตอร์ต่อกับอินเตอร์เน็ตสามารถดาว์นโหลดโปรแกรมไคลแอนท์ของ SETI@Home ไปติดตั้งได้สำหรับ Windows และ Mac โปรแกรมไคลเอนท์ จะทำหน้าที่แทน screensaver คือจะประมวลผลเมื่อผู้ใช้ไม่ได้ใช้เครื่องอยู่ ส่วนบน platform อื่นไคลแอนท์จะทำงานเป็น background mode หรือ ควาสำคัญในระดับต่ำ (low priority) เพื่อไม่ให้รบกวนงานอื่นๆที่กำลังทำอยู่ คือจะไม่รบกวนงานปกติของผู้ใช้ปัจจุบัน SETI@Home มีกำลังในการประมวลผลอยู่ที่ประมาณ 180 TFLOPs/sec โดยมีการเขียนซอฟท์แวร์สำหรับทำงานแบบ ออฟไลน์(offline) ได้ด้วย จึงสามารถทำงานได้แม้จะไม่ได้ต่อกับ

อินเตอร์เน็ตโดยเครื่องจะทำการส่งผลลัพธ์ที่เสร็จแล้วกลับไปเมื่อเราทำการเชื่อมต่ออินเตอร์เน็ตในภายหลัง



รูปที่ 2-7 Arecibo Radio Scope ที่ประเทศ Puerto Rico ใช้สำหรับการรับฟังสัญญาณของ SETI@Home

2.9.2. Teragrid

Teragrid เป็นโครงการของมูลนิธิวิทยาศาสตร์แห่งชาติ (National Science Foundation หรือ NSF) ของสหรัฐอเมริกา เพื่อใช้สำหรับการใช้งานเพื่อการค้นคว้าในหลายๆสาขา โปรแกรมนี้มีกำลังในการประมวลผล 40 TeraFLOPs/s และมีอุปกรณ์จัดเก็บที่มีขนาดประมาณ 2 Petabyte ครอบคลุมทั่วทั้งศูนย์คอมพิวเตอร์ของมหาวิทยาลัยอิลลินอยส์, มหาวิทยาลัยซานดิเอโก, ห้องปฏิบัติการแห่งชาติอาร์กอนและแคลเทค (Argonne National Labs and Cal Tech) เชื่อมต่อกันโดยผ่านท่อ fiber optic ของ Qwest ที่มีความเร็วในการเชื่อมต่ออยู่ที่ 40 Gbps

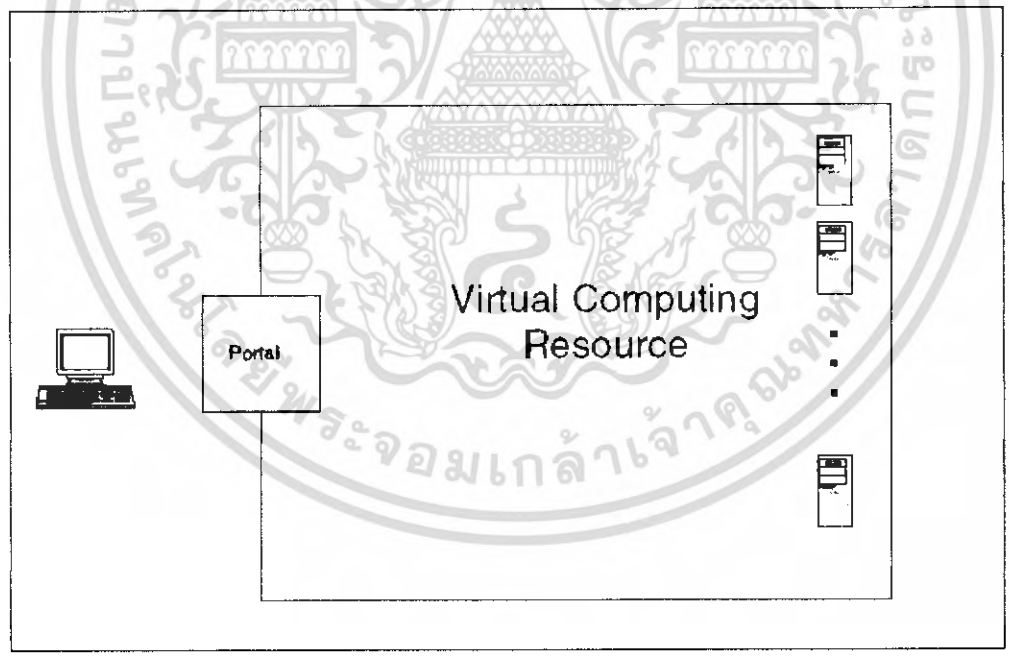
บทที่ 3
องค์ประกอบของระบบกริด(Grid Component) และ GLOBUS toolkit

3.1. องค์ประกอบของระบบกริด(Grid Component)

3.1.1. portal – user interface

เปรียบเสมือนเป็นกำแพงชั้นแรกในการเข้าใช้ระบบกริดผู้เข้าใช้ จะมองไม่เห็นการทำงานภายในระบบเป็นส่วนของการติดต่อระหว่างผู้ใช้ระบบกริดและตัวระบบ เพื่อให้ผู้ใช้สามารถเรียกใช้แอพพลิเคชั่นขึ้นมาทำงานได้ ในมุมมองของผู้ใช้ระบบกริดจะมองระบบเป็นเหมือนแหล่งทรัพยากรเสมือน ซึ่งใน globus toolkit จะไม่มีส่วนของพอร์ทอล (Portal) นี้ดังจะเห็นได้จากรูปที่ 3-1 หากต้องการเรียกใช้จะต้องใช้เครื่องมืออื่นๆเข้ามาช่วย อาทิเช่น Websphere

พอร์ทอลอาจจะนำเสนอในรูปแบบของ เว็บเพจ(web page) ซึ่งเป็นรูปแบบที่ง่ายในการจัดการหรือติดต่อกับแอพพลิเคชั่นของระบบกริด ซึ่งในพอร์ทอลนี้จะมีส่วนของการพิสูจน์ตน ,การเสนองาน , การดูแลงานที่ทำอยู่ และการแสดงผลของงานนั้นๆ ส่วนที่มีความสำคัญมากที่สุดของพอร์ทอล คือทำอย่างไรให้ผู้ใช้ระบบได้รับความสะดวกในการใช้งาน



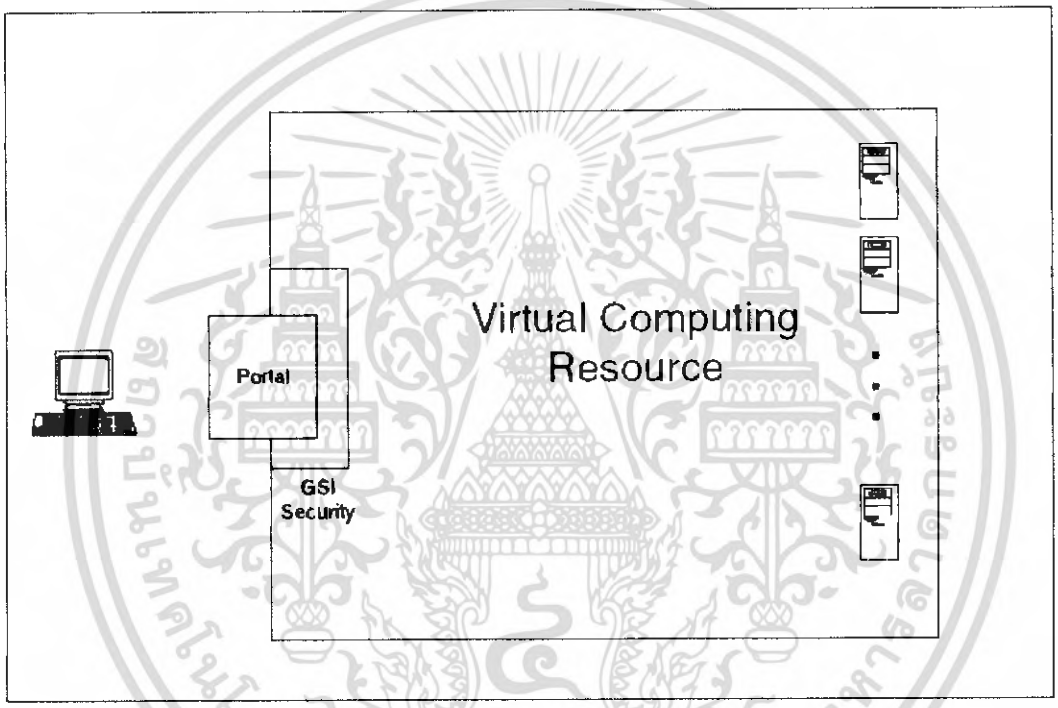
รูปที่ 3-1 ตำแหน่งของพอร์ทอลในระบบกริด

3.1.2. Security (ระบบรักษาความปลอดภัย)

องค์ประกอบที่สำคัญอย่างหนึ่งคือเรื่องของความปลอดภัย จะใช้บริการหลัก 3 บริการด้วยกัน คือ การพิสูจน์ตัวตน , การพิสูจน์สิทธิ์ , การเข้ารหัสลับ ซึ่งใน globus จะเรียกองค์ประกอบนี้ว่า GSI ดัง

แสดงให้เห็นในส่วนที่แรเงาในรูปที่ 3-2 ซึ่งใน GSI จะใช้ OpenSSL ในการสร้าง ตลอดจนมีกลไกของซิงเกิ้ลไซน์ออน

ในระบบกริด เมื่อต้องการเรียกใช้ทรัพยากรด้วยการทำงานกับระบบที่อยู่ระยะไกล จะต้องมั่นใจได้ว่าการทำงานแบบนี้จะไม่มีใครได้ล่วงรู้ข้อมูลที่กำลังทำการส่งอยู่ และหากเครื่องที่ปฏิบัติงานอยู่เป็นผู้ให้บริการด้านทรัพยากร จะต้องมั่นใจว่าถ้ามีผู้ใช้ระบบเข้ามาทำงานที่เครื่องนี้ จะไม่ทำให้เกิดการรบกวนเจ้าของเครื่อง หรือแม้กระทั่งสามารถเข้าถึงข้อมูลของระบบได้ ซึ่งระบบรักษาความปลอดภัยใน globus จะประกอบไปด้วยการพิสูจน์ตน การพิสูจน์สิทธิ์ และการติดต่อกันระหว่างทรัพยากรอย่างปลอดภัย



รูปที่ 3-2 ตำแหน่งของ GSI ในระบบกริด

- *การพิสูจน์ตน* สามารถเรียกใช้ซิงเกิ้ลไซน์ออนโดยการใช้คำสั่ง grid-proxy-init เพื่อสร้างพรอกซี่ ชั่วคราวโดยการสร้างพรอกซี่ นี้จะใช้คีย์ส่วนตัวของผู้ที่ทำการสร้างพรอกซี่ เมื่อสร้างเซสชั่นแล้ว เป็นหน้าที่ของเครื่องเซิร์ฟเวอร์ที่จะตัดสินใจในการพิสูจน์สิทธิ์ของผู้ใช้ระบบคนนั้น พรอกซี่ จะถูกสร้างก่อนที่ผู้ใช้ระบบจะสามารถทำงานหรือส่งข้อมูลได้
- *การพิสูจน์สิทธิ์* เมื่อทำการพิสูจน์ตนแล้ว จะต้องทำการพิสูจน์สิทธิ์เป็นลำดับต่อไป เมื่อมีการร้องขอการทำงานใดๆผ่าน พรอกซี่ ซึ่งใน พรอกซี่ จะมีชื่อของผู้ใช้ที่เรียกว่า DN จะต้องทำการตรวจสอบชื่อ DN ให้ตรงกับ ชื่อผู้ใช้ระบบที่เป็น local

- *การติดต่อระหว่างกันภายในกริด* ถ้าหากว่าต้องการเข้าถึง เครื่องเซิร์ฟเวอร์หรือเครื่องไคล์เอนท์อาจ จะต้องเรียกใช้การติดต่อแบบเครือข่ายระยะไกล เพื่อการเข้าใช้ระบบกริด ตัวอย่างเช่นใช้การ telnet ย่อมไม่อาจมั่นใจได้ว่าจะไม่ถูกขโมยข้อมูลออกไปจากระบบเครือข่าย ซึ่งอาจจะใช้ Secure Shell (SSH) เข้ามาช่วยทำให้เกิดความปลอดภัยมากขึ้น ซึ่งใน globus มีบริการนี้อยู่แล้วแต่จะต้องเปิดบริการให้เรียบร้อยก่อน

3.1.3. Broker ,Grid Information Service , Monitoring and Discovery Service (MDS)

เมื่อทำการพิสูจน์ตัวตนเรียบร้อยแล้วผู้ใช้ระบบกริดจะทำการเรียกใช้แอพพลิเคชั่น และลำดับต่อไปที่จะต้องทำคือ การระบุและแยกแยะทรัพยากรที่เหมาะสมให้กับผู้เรียกใช้แอพพลิเคชั่นนั้น และนี่เป็นหน้าที่รับผิดชอบของ broker ซึ่งใน globus จะไม่มี broker แต่จะมีโปรโตคอล LDAP ที่ช่วยจัดการในเรื่องของบริการ ซึ่งเรียกบริการนั้นว่า Grid Information Service(GIS) หรืออาจจะเรียกว่า Monitoring and Discovery Service(MDS) ซึ่งบริการเหล่านี้จะให้ข้อมูลเกี่ยวกับทรัพยากรที่ผู้ใช้ระบบสามารถเรียกใช้ได้

3.1.3.1. GIS (Grid Information Service)

เป็นองค์ประกอบที่มีความสำคัญมากเพราะเก็บข้อมูลเกี่ยวกับทรัพยากร ขนาด และการใช้ประโยชน์ภายในกริด ซึ่งโดยปกติแล้วการใช้ประโยชน์ของหน่วยประมวลผล (ซีพียู) และทรัพยากรในกริดจะแกว่งขึ้นลงทั้งนี้ขึ้นอยู่กับโพรเซสและการใช้ข้อมูลร่วมกันซึ่งในระบบกริดจะมี grid information serviceเป็นบริการที่ให้ข้อมูลและทำการอัพเดทสถานะของทรัพยากรภายในกริดซึ่งเครื่องไคล์เอนท์ brokerทำหน้าที่จัดการทรัพยากรภายในกริด จะใช้ข้อมูลจาก GIS เหล่านี้ในการตัดสินใจเลือกทรัพยากรให้เหมาะสมกับงาน

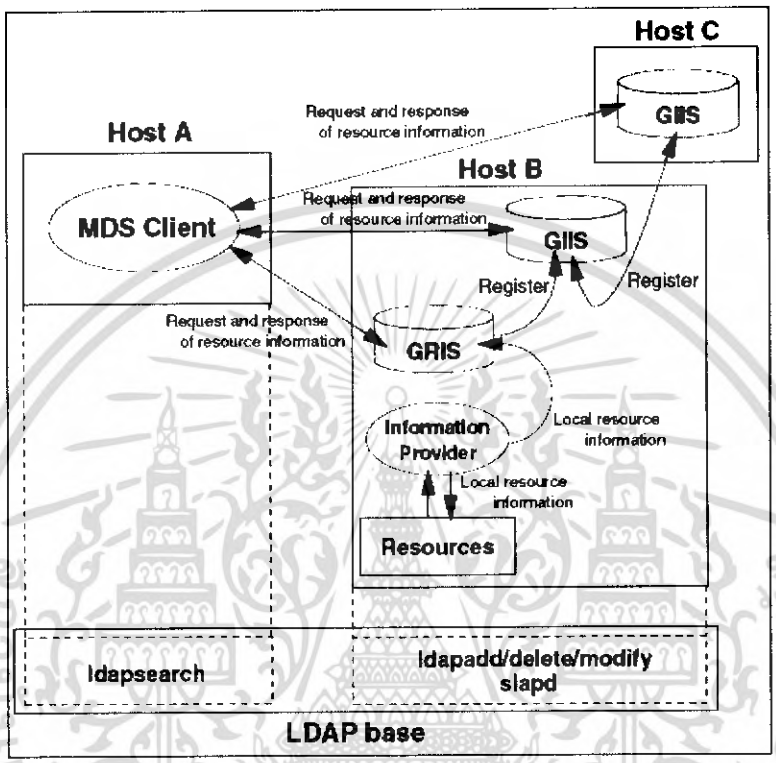
3.1.3.2. MDS (Monitoring and Discovery Service)

MDS ดังแสดงให้เห็นจากส่วนที่แรงเงาในรูปที่ 3-4 ทำหน้าที่ช่วยในการเข้าถึงข้อมูลทั้งข้อมูลสถิต และ พลวัตน์ ของทรัพยากรในระบบกริด ซึ่งใช้มาตรฐานของโปรโตคอล LDAP (Lightweight Directory Aeccss Protocol) องค์ประกอบของ MDS คือ

- **GRIS (Grid Resource Information Service)** เป็นแหล่งเก็บข้อมูลของทรัพยากรที่ได้รับมาจาก Information Provider GRIS สามารถลงทะเบียนข้อมูลที่มีอยู่กับ GIIS และข้อมูลของทรัพยากรนั้นจะมีเวลาอยู่ใน GRIS ในช่วงเวลาที่จำกัด คือมี TTL (Time-to-live) หาก GRIS ไม่ได้รับการร้องขอทรัพยากรเหล่านั้นเป็นเวลาเท่ากับ TTL แล้ว GRIS จะลบข้อมูลเหล่านั้นทิ้งไป และหากมีการร้องขอข้อมูลเกี่ยวกับทรัพยากรเหล่านั้นมา GRIS จะเรียกข้อมูลที่สัมพันธ์กับลักษณะข้อมูลที่ร้องขอมา และใหม่ล่าสุดจาก information provider
- **GIIS (Grid Index Information Service)** เป็นแหล่งเก็บดัชนีของข้อมูลทรัพยากร ซึ่งจะได้รับการลงทะเบียนจาก GRIS หรือ GIIS ที่อื่นๆ ลักษณะของกลไกของ GIIS จะเป็นโครงสร้างระดับชั้น คล้ายกับ DNS และแต่ละ GIIS จะมีชื่อเรียกเป็นของตัวเอง จะทำให้

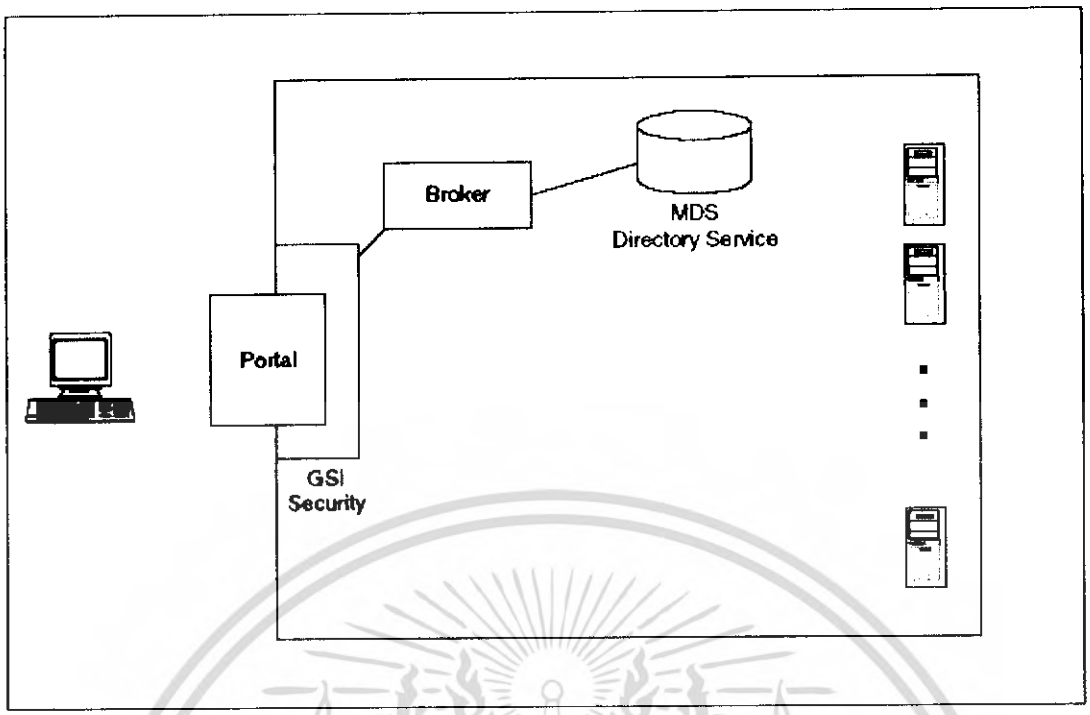
เครื่องไคล์เอนท์ที่เป็นผู้ใช้งานระบบกริดจะสามารถระบุชื่อและโหนดของ GIIS เพื่อค้นหาข้อมูลได้

- **Information Providers** ทำหน้าที่แปลงคุณสมบัติและสถานะของทรัพยากรให้อยู่ในรูปแบบที่ได้รับการกำหนดไว้
- **MDS Client** จะใช้รูปแบบเดียวกับ คำสั่ง LDAP ของเครื่องไคล์เอนท์



รูปที่ 3-3 ภาพการทำงานของ MDS

จากรูปที่ 3-3 เป็นภาพรวมการทำงานของ MDS ส่งข้อมูลทรัพยากรที่ได้มาจาก Information Provider ไปยัง GRIS GRIS ทำหน้าที่ลงทะเบียนให้ข้อมูลที่ได้รับมาเป็นข้อมูลทรัพยากรของตน (local) และลงทะเบียนข้อมูลนี้ที่ GIIS และ GIIS จะทำหน้าที่ลงทะเบียนข้อมูลนี้กับ GIIS ตัวอื่นๆ และทำเช่นนั้นต่อไปเรื่อยๆ MDS client จะสามารถค้นหาข้อมูลของทรัพยากรได้โดยตรงจาก GRIS ถ้าเป็นข้อมูลของตน (local) และค้นหาข้อมูลได้จาก GIIS ถ้าเป็นข้อมูลที่อยู่แหล่งทรัพยากรอื่น (grid-wide)



รูปที่ 3-4 ตำแหน่งของBroker และ MDS ในระบบกริด

3.1.4. Scheduler

เมื่อทรัพยากรได้รับการระบุตัวตนแล้ว อันดับต่อไปจะต้องสร้าง schedule ให้กับทรัพยากรเหล่านั้น เพื่อให้ทำงานได้ตามที่ต้องการ เมื่อมีกลุ่มของงานที่สามารถทำงานได้โดยไม่ต้องเกี่ยวเนื่องกับงานอื่น จะไม่มีความต้องการ scheduler เพื่อทำหน้าที่ในการแบ่งงาน แต่หากไม่เป็นเช่นนั้น จะต้องการ Scheduler และ inter-process communication มาช่วยในการคำนวณงานชิ้นนั้น

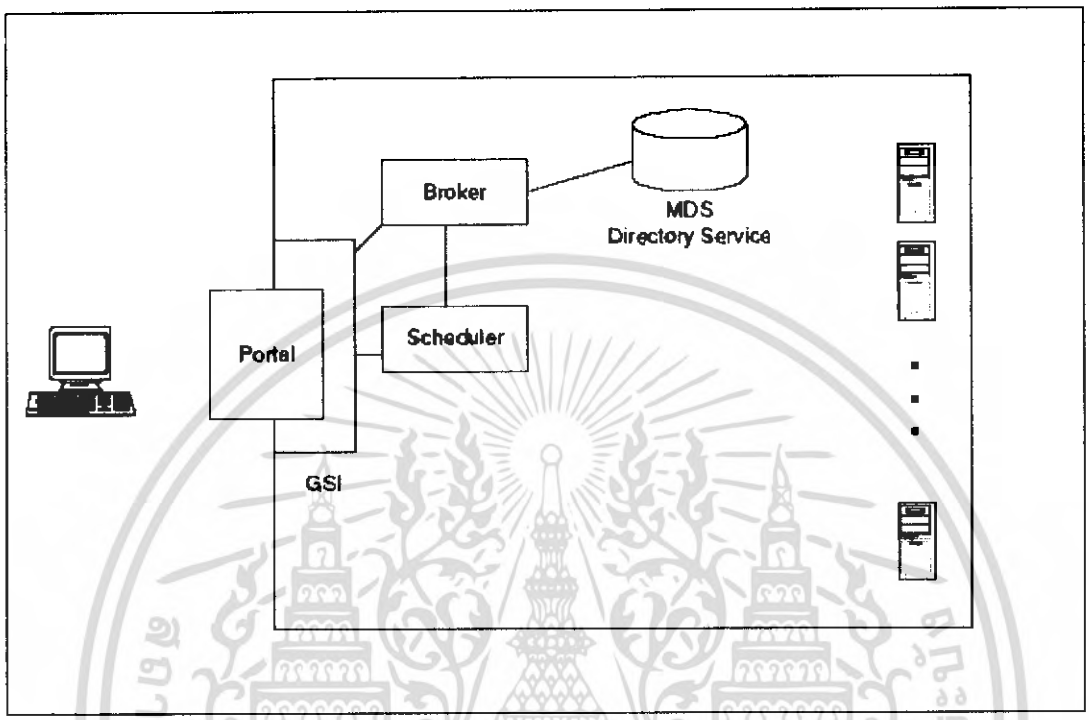
จะต้องทราบว่าในระบบกริดจะมี scheduler หลายระดับชั้น ซึ่ง scheduler ที่มีระดับชั้นสูงๆ เรียกว่า meta scheduler ใน globus toolkit จะไม่มี scheduler ให้ แต่หากต้องการให้มีจะต้องเรียกใช้ผลิตภัณฑ์อื่นๆ เช่น Condor-G

Condor-G

Condor เป็นซอฟแวร์ที่ประกอบไปด้วย 2 ส่วนคือ ส่วนที่ทำหน้าที่จัดการทรัพยากร และ ส่วนที่ทำหน้าที่จัดการงาน

- ส่วนที่ทำหน้าที่จัดการทรัพยากร จะช่วยทำให้เครื่องที่ทำงานอยู่สามารถทำงานได้อย่างมีประสิทธิภาพสูงสุด
- ส่วนที่ทำหน้าที่จัดการงาน เป็นส่วนที่เสนองานใหม่ให้กับระบบ หรือหยุดการทำงานแต่ยังคงเก็บค่านั้นอยู่ ตลอดจนมีข้อมูลเกี่ยวกับการจัดเรียงงาน และสถานะของงานที่ทำเสร็จเรียบร้อยแล้ว

สำหรับ Condor-G จะใช้ globus toolkit ในการเริ่มต้นทำงานที่เครื่องคอมพิวเตอร์ที่อยู่ระยะไกลแทนที่จะใช้โปรโตคอล condor ประโยชน์ของการใช้ Condor คือความสามารถในการเสนองานได้หลายงานในเวลาเดียวกัน โดยใช้วิธีการของคิว และการดูแลระบบของงานที่ถูกเสนอไปแล้วด้วยรูปแบบที่ condor มีอยู่แล้ว



รูปที่ 3-5 ตำแหน่งของ Scheduler ในระบบกริด

3.1.5. Data Management (การจัดการข้อมูลในระบบกริด)

เมื่อมีการติดต่อกับภายในกริด สิ่งหนึ่งที่ต้องให้ความสำคัญคือ การเคลื่อนย้ายข้อมูลหรือไฟล์ระหว่างแต่ละจุดในระบบ ทั้งนี้ต้องให้การเคลื่อนย้ายที่เกิดขึ้นมีความเชื่อถือได้และมีความปลอดภัย ซึ่งใน globus จะมีบริการที่ทำหน้าที่นี้ คือ Grid access to Secondary Storage (GASS) อาทิเช่น GridFTP สร้างขึ้นบนมาตรฐานของ FTP โปรโตคอล และเพิ่มฟังก์ชั่นบางอย่างเพื่อให้สามารถทำงานร่วมกับ GSI ที่มีอยู่แล้วได้

- **GridFTP**

  GridFTP เพิ่มความสามารถจาก โปรโตคอล FTP เดิมคือ

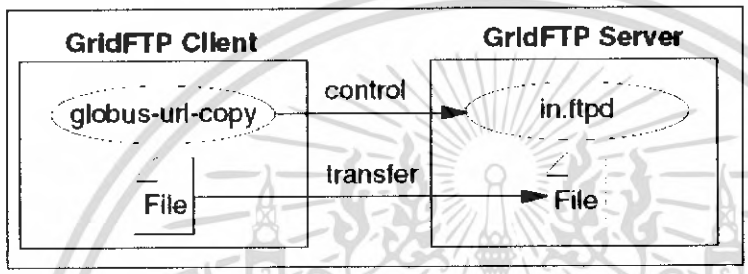
  1. สนับสนุนการใช้งานคู่กับ GSI ของ globus toolkit และ Keberos และสามารถใช้การพิสูจน์ตัวตนได้ทั้งสองแบบ ผู้ใช้ระบบกริดสามารถเลือกระดับของความถูกต้องของข้อมูล และการรักษาความลับให้แตกต่างกันได้
  2. อนุญาตให้มีการส่งไฟล์ระหว่างเครื่องเซิร์ฟเวอร์ได้ (Third-party data transfer)

3. สามารถส่งข้อมูลแบบขนานได้โดยการใช้สายสตรีมของ TCP ได้หลายสายซึ่งจะสามารถรวมแบนด์วิดธ์ได้
4. ส่งไฟล์บางส่วนได้ คือแบ่งแยกไฟล์และทำการส่ง
5. มีการส่งข้อมูลที่เชื่อถือได้ เมื่อเกิดความผิดพลาดสามารถกู้คืนข้อมูลได้
6. สามารถกำหนดขนาดของบัฟเฟอร์ของโปรโตคอล TCP ได้

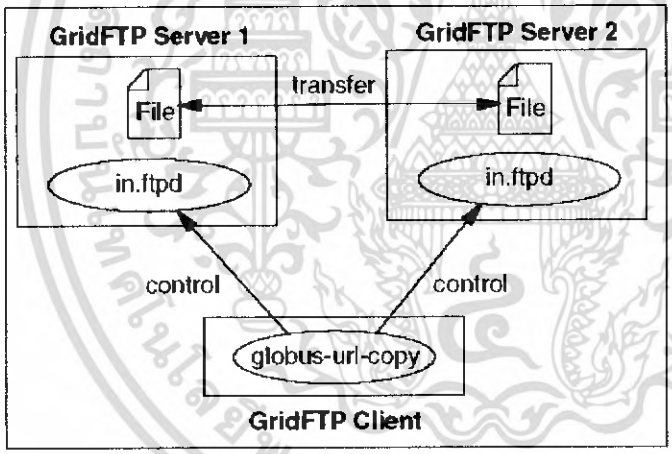
ขั้นตอนการทำงาน ใน Globus toolkit มีส่วนของ GridFTP ของเครื่องเซิร์ฟเวอร์และเครื่องไคล์เอนท์ ซึ่งสร้างขึ้นโดย in.ftpd daemon และใช้ร่วมกับคำสั่ง globus-url-copy

เครื่องเซิร์ฟเวอร์และเครื่องไคล์เอนท์ของ GridFTP จะสนับสนุนการส่งข้อมูล 2 ลักษณะคือ

- Standard



- Third Party

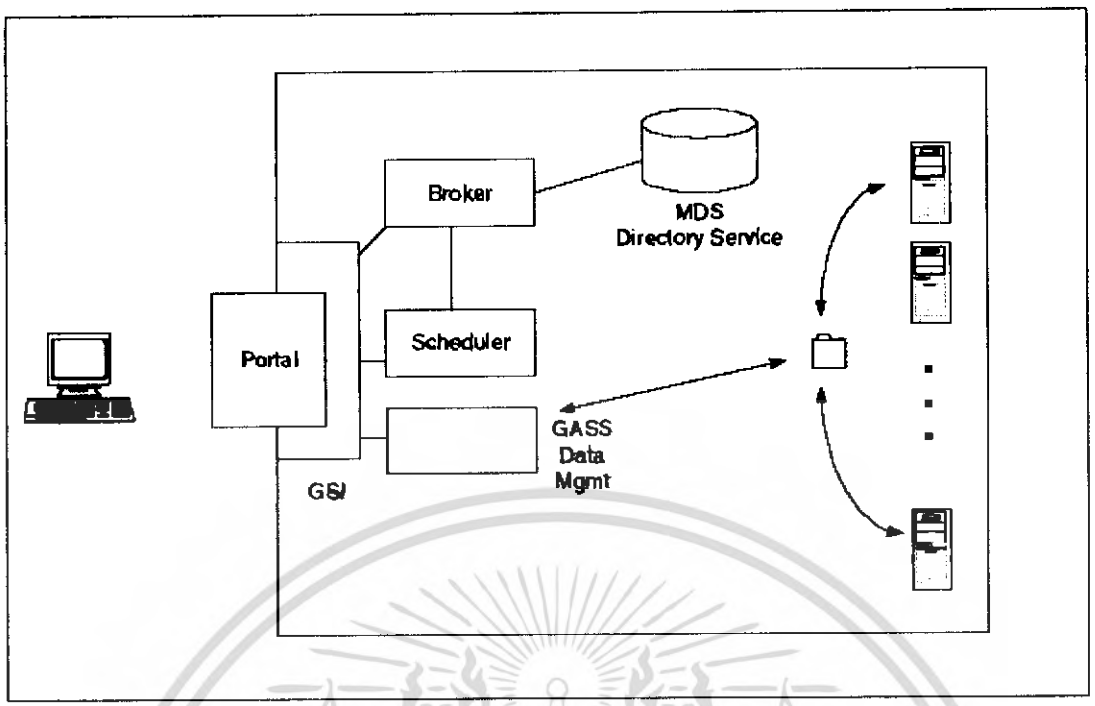


- **Globus Access to Secondary Storage (GASS)**

GASS ใช้ในการส่งข้อมูลระหว่าง GRAM ของเครื่องไคล์เอนท์ และ GRAM ของเครื่องเซิร์ฟเวอร์ GASS ยังทำหน้าที่ในการเปิด , ปิด หรือเลือกข้อมูลจากกลุ่มของข้อมูลในระบบกริด GASS ช่วยกำจัดความต้องการที่จะเข้าไปยังแหล่งข้อมูลนั้น และส่งไฟล์ ตลอดจนการติดตั้งไฟล์ที่ทำหน้าที่ในการกระจายระบบ

- **Replica Management**

สามารถเก็บสำเนาของแหล่งเก็บข้อมูลไว้ที่แหล่งเก็บข้อมูลของเราเองเพื่อให้เกิดความรวดเร็วในการเข้าถึงข้อมูลที่เราต้องการ



รูปที่ 3-6 ตำแหน่งของ GASS ในระบบกริด และการใช้ GridFTP

3.1.6. Job and Resource Management (การจัดการงานและทรัพยากรในระบบกริด)

บริการที่ทำหน้าที่เป็นส่วนหลักของการทำงานในระบบกริด คือ Resource Allocation Manager (GRAM) ทำหน้าที่ในการดูแลทรัพยากรและจัดการกับทรัพยากรโดยตรง โดยการตรวจสอบสถานะของทรัพยากร และ รับผลเมื่อทำงานนั้นสำเร็จ

เมื่อเครื่องไคล์เอนท์ต้องการทำงาน จะส่งคำขอร้องไปยังเครื่องที่อยู่ระยะไกล และจะมีผู้ดูแล คือ gatekeeper ซึ่ง gatekeeper จะสร้างผู้ดูแลงานเหล่านั้นและเมื่องานนั้นเสร็จสิ้นลง ผู้ดูแลงานนั้นจะจะส่งรายงานสถานะไปยังเครื่องไคล์เอนท์ และทำการสิ้นสุดงานนั้น

GRAM จะประกอบไปด้วยระบบย่อยๆ คือ

- ***คำสั่ง globusrun*** ซึ่งใช้ติดต่อกับแอพพลิเคชั่น เป็นคำสั่งที่ใช้ในการเสนองาน โดยการติดต่อกับทรัพยากร คำสั่งนั้นจะถูกส่งไปในรูปแบบของ RSL สตริง ที่ระบุพารามิเตอร์และคุณสมบัติอื่นๆที่ต้องใช้ในการทำงานให้สำเร็จ
- ***Resource Specification Language (RSL)*** คือภาษาที่เครื่องไคล์เอนท์ใช้ในการระบุงานที่ต้องการทำ ในการร้องขอและเสนองานจะใช้ RSL ในการอธิบายความต้องการของเครื่องไคล์เอนท์ เช่นไฟล์ที่ต้องใช้ในการทำงาน,พารามิเตอร์ และข้อมูลอื่นๆที่เกี่ยวกับการทำ redirection ไปยัง stdin , stdout, stderr โดยปกติแล้วจะมีมาตรฐานในการระบุทุกๆข้อมูลที่ต้องการในการทำงาน ซึ่งไม่ขึ้นกับองค์ประกอบอื่นๆ ซึ่งจะเป็น

หน้าที่ของผู้จัดการงาน ในการแบ่งแยกข้อมูลที่ได้รับ และเริ่มต้นทำงานในแนวทางที่เหมาะ สม

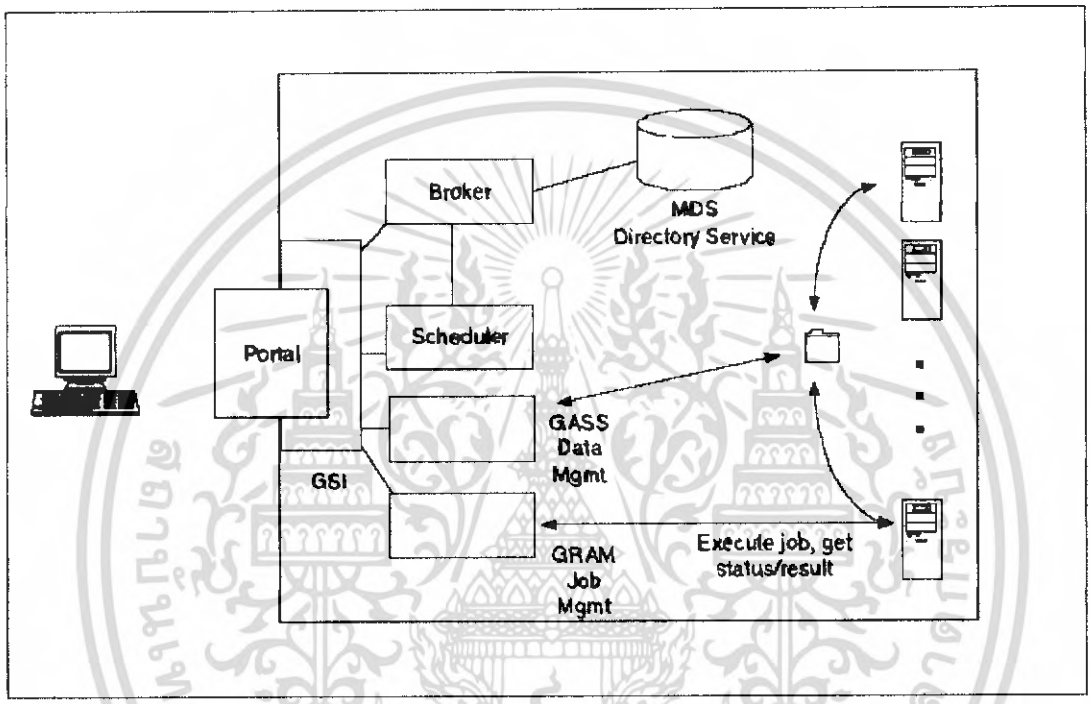
- ***Gatekeeper Daemon*** เป็นตัวจัดการการติดต่อที่มีกลไกการรักษาความปลอดภัยระหว่างเครื่องเซิร์ฟเวอร์และเครื่องไคล์เอนท์ gatekeeper จะใช้ GSI ในการพิสูจน์ตัวตนก่อนที่จะสามารถทำงานได้ หลังจากการพิสูจน์ตัวตนสำเร็จ gatekeeper จะเริ่มต้นทำงานจริง และจะให้อำนาจกับเครื่องไคล์เอนท์ในการติดต่อกับระบบกริด
- ***ผู้จัดการงาน (job manager)*** ถูกสร้างโดย gatekeeper daemon ซึ่งเป็นส่วนหนึ่งของกระบวนการร้องขอให้ทำงาน ผู้จัดการงานจะมีส่วนที่ติดต่อคือส่วนของการควบคุมการจองทรัพยากร ซึ่งอาจจะอยู่ในบริการที่เรียกว่า Job Scheduler หน้าที่การทำงานของผู้จัดการงานคือ
  a) ทำการแยกแยะ RSL String ที่ส่งมาจากเครื่องไคล์เอนท์
  b) ทำการจองทรัพยากรให้เหมาะสมกับงานที่ร้องขอมา
  c) ส่งข้อมูลกลับไปหาเครื่องไคล์เอนท์ (ในกรณีที่จำเป็น)
  d) ได้รับสถานะจากเครื่องไคล์เอนท์ คือ สถานะร้องขอ หรือ สถานะที่ต้องการยกเลิก
  e) ส่งผลลัพธ์ไปยังเครื่องไคล์เอนท์โดยการใช้ GASS
- ***Dynamically-Updated Request Online Coallocator (DUROC)*** เป็นแอพพลิเคชั่นที่อนุญาต ให้ผู้ใช้ระบบกริดสามารถรับงานหลายงานได้ด้วยคำสั่งเดียว DUROC จะใช้ coallocator เพื่อใช้ในการคำนวณงานและจัดการงานผ่านผู้จัดการทรัพยากรหลายๆ แหล่ง ใน globus สามารถเรียกใช้ DUROC ด้วยคำสั่ง globus-duroc RSL script จะมีส่วนประกอบที่มีความหมายของ DUROC ด้วย ซึ่ง GRAM ที่เครื่องไคล์เอนท์จะทำการแยกแยะความหมายของ DUROC และทรัพยากรนั้นจะถูกจองด้วยการทำงานของผู้จัด การงานหลายตัว

การพิจารณาแอพพลิเคชั่นเมื่อเป็นมุมมองของการจัดการทรัพยากร

- การเลือกทรัพยากรที่เหมาะสมกับแอพพลิเคชั่นbroker ทำหน้าที่เลือกทรัพยากรที่เหมาะสมและเป็นไปตามความต้องการของแอพพลิเคชั่น และผู้พัฒนาแอพพลิเคชั่นจะต้องสามารถกำจัด ข้อจำกัดบางอย่างของแอพพลิเคชั่นได้ เพื่อให้เป็นการสร้างโอกาสที่เพิ่มขึ้นที่จะจองทรัพยากรที่ว่างอยู่ได้มากขึ้น
- งานย่อยๆจำนวนมาก หากแอพพลิเคชั่นมีงานหลายงาน ผู้ออกแบบจะต้องเข้าใจความสัมพันธ์และความเกี่ยวเนื่องกายในงานย่อยๆเหล่านั้นจึงต้องหาตรรกะเพื่อมารองรับข้อจำกัดเหล่านี้ อันได้แก่
  a) Inter-process Communication
  b) การใช้ข้อมูลร่วมกัน

c) การทำงานขนานกันของงาน

- การเข้าถึงผลลัพธ์ ถ้างานเหล่านั้นได้ส่งค่าของสถานะ หรือผลลัพธ์จำนวนหนึ่งออกมา แอพพลิเคชั่นสามารถจะรับข้อมูลเหล่านั้นได้จาก stdout และ stderr
- ผู้จัดการงาน GRAM มีกลไกในการค้นหาสถานะของงาน แอพพลิเคชั่นต้องใช้ความสามารถของผู้จัดการงาน เพื่อบอกให้ผู้ใช้ระบบทราบได้ว่าควรจะทำอย่างไรกับทรัพยากร เช่น เมื่องานๆหนึ่งผิดพลาด งานอื่นๆที่มีความเกี่ยวเนื่องกับงานชิ้นนี้ย่อมต้องยกเลิกด้วย และทำให้ต้องทำการปล่อย (free) ทรัพยากรนี้นั่นเอง



รูปที่ 3-7 ตำแหน่งของ GRAM ในระบบกริด

3.2. GLOBUS Toolkit (GT3)

3.2.1. GT3 คืออะไร

GT3 (Globus Toolkit 3) นั้นเป็นเครื่องมือที่พัฒนาโดยภาษา Java ที่นำคุณสมบัติของ OGSI มาใช้ในการพัฒนา กล่าวคือ GT3 เปรียบเสมือนเฟรมเวิร์คที่ใช้ในการสร้างกริดเซอร์วิส

3.2.2. เซอร์วิสพื้นฐานที่จัดเตรียมโดย GT3

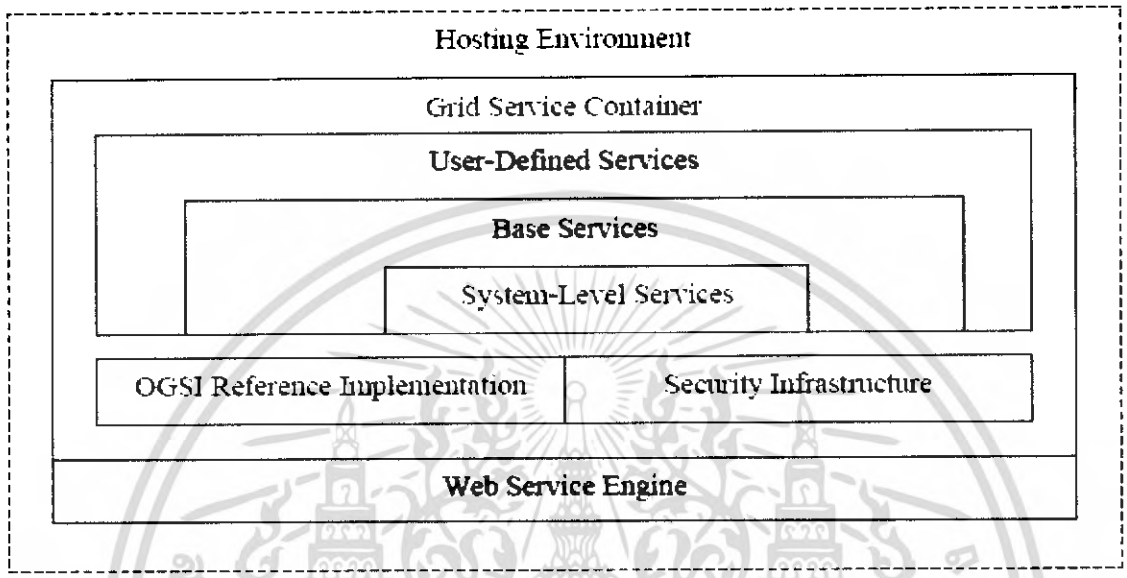
GT3 นั้นได้จัดเตรียมเซอร์วิสพื้นฐานไว้มากมาย ซึ่งแบ่งออกเป็นส่วนๆตามหน้าที่ ได้ดังต่อไปนี้

- Core (แกน)
- Security (ด้านความปลอดภัย) อันได้แก่ GSI และ CAS
- Data Management (การจัดการข้อมูล) อันได้แก่ GridFTP, RLS, RFT, XIO

- Resource Management (การจัดการทรัพยากร) อันได้แก่ GRAM
- Information service (การให้บริการด้านข้อมูล) อันได้แก่ MDS

3.2.2.1. แกน (Core)

ภาพด้านล่างนี้ เป็นภาพโครงสร้างของ GT3 ซึ่ง องค์ประกอบของ Core จะถูกแทนในช่องสีขาว



รูปที่ 3-8 โครงสร้างของ GT3

3.2.2.1.1. Hosting Environment

คือ กลุ่มโฮสต์ที่ใช้ในการสร้างระบบกริด ได้จำแนกไว้ 4 ประเภทที่แตกต่างกัน คือ

1. Embeddedโดยจะมีประโยชน์กับไคล์เอนท์หรือเซิร์ฟเวอร์ขนาดเล็กที่ให้ใช้งานกริดเซอร์วิส
2. Standalone มักใช้กับโปรแกรมขนาดเล็ก
3. Servlet ใช้พื้นฐานของ Java Servlet Engine มาเป็นมาตรฐานของ Container
4. EJB (Enterprise Java Bean) ใช้ EJB application server มาช่วยในการทำระบบกริด

ซึ่งทุกประเภทของ Hosting environment ที่กล่าวไปข้างต้น สามารถรองรับการทำกริดเซอร์วิสได้ โดยแบบที่ 3 นั้นจะเป็นการสร้างกริดเซอร์วิสแบบชั่วคราวจริงๆ แต่แบบที่ 4 จะชั่วคราวที่การทำงานในส่วนของ EJB

3.2.2.1.2. Grid Service Container

เป็นส่วนที่จะใช้เก็บเซอร์วิสต่างๆ ได้แบ่งการทำงานออกเป็น 3 ประเภท ดังนี้

1. คิดเกี่ยวกับเซอร์วิสขนาดเล็กๆ โดยใช้การค้นหา 2 แบบ คือ pull และ push
2. ควบคุมซอฟท์สเตทของเซอร์วิสอินสแตนซ์ที่ทำงานเป็นสเตทอย่างเต็มรูปแบบซึ่งอ้างได้อย่างทั่วถึง

3. ขนส่งส่วนที่เป็นอิสระกับโครงสร้างการทำงานที่เน้นความปลอดภัยตาม Grid Service Infrastructure โดยใช้ delegation, message signing และ encryption มาใช้ในการอนุญาต

3.2.2.1.3. OGSI Reference Implementation

เป็นส่วนที่ทำมาจาก OGSI ซึ่งเป็นนำคุณสมบัติทั้งหมดตามที่กำหนดไว้ใน OGSI มาช่วยในการสร้างระบบ เช่น GridService, Factory, Notification (Source/Sink/Subscription), HandleResolver, ServiceGroup (Entry/Registration)

3.2.2.1.4. Security Infrastructure

เป็นส่วนที่ใช้เกี่ยวกับความปลอดภัยในระดับการส่งข้อความ โดยจะมีองค์ประกอบของความปลอดภัยระดับข้อความ, การรับรองสิทธิ และ การอนุญาตขึ้นกับกริดแม็พ ซึ่งสิ่งที่นำมาช่วยในข้อนี้ก็มี WS-Security, XML-Signature และ XML-Encryption

3.2.2.1.5. System Level Services

เป็นกริดเซอร์วิสที่ทำตาม OGSI ซึ่งสามารถใช้งานได้โดยโดยกริดเซอร์วิสตัวอื่นๆทั้งหมด ซึ่ง GT3 ในตอนนี้ ได้แบ่งเซอร์วิสส่วนนี้ออกเป็น 3 ระดับ

- Ping service: ใช้ในการ "ping" ไปยัง hosting environment เพื่อเช็คว่าเข้าถึงได้หรือไม่
- Logging Management Service: ยอมให้แก้ไขการกรอง log และใช้สร้าง log ที่คงอยู่เพื่อช่วยในการจัดการเวลาทำงาน
- Management Service: เป็นส่วนจัดเตรียมการมอนิเตอร์สถานะและการโหลดของกริดเซอร์วิสคอนเทนเนอร์ และเพื่อใช้ปิดคอนเทนเนอร์ และยังเป็นส่วนที่ยอมให้สั่งเปิดและปิดเซอร์วิสอินสแตนซ์

3.2.2.2. ด้านความปลอดภัย (GSI และ CAS)

3.2.2.2.1. GSI (Grid Security Infrastructure)

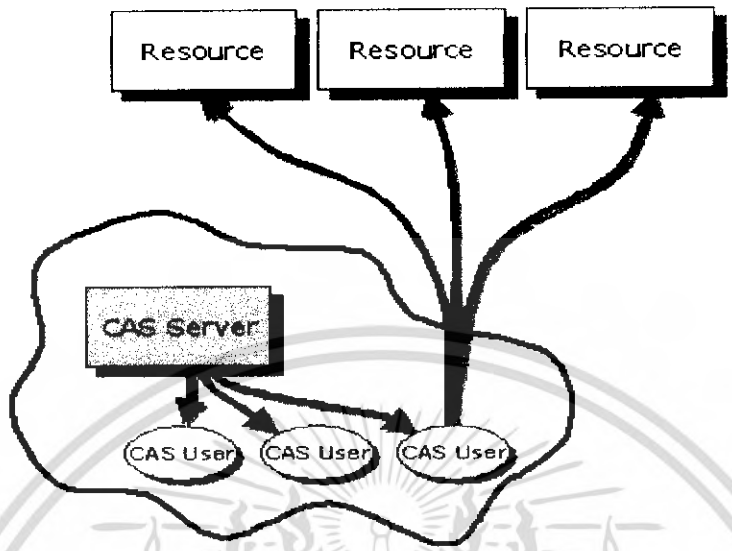
Grid Security Infrastructure (GSI) เป็นการเข้ารหัสคีย์สาธารณะ เพื่อใช้ประโยชน์ในด้านความปลอดภัยต่างๆ โดยจุดประสงค์เบื้องต้นที่ทำให้เกิด GSI ขึ้นนั้น คือ

- ความต้องการการสื่อสารที่ปลอดภัย รับรองได้ บางทีต้องเป็นความลับระหว่างการสื่อสารที่ใช้ระบบกริด
- ความต้องการที่จะรองรับการใช้งานข้ามเขตของแต่ละองค์กร ภายใต้ระบบความปลอดภัยที่ควบคุมจากศูนย์กลาง
- ความต้องการที่จะรองรับ "single sign-on" สำหรับผู้ใช้งานกริด โดยที่การสื่อสารนั้น จะครอบคลุมไปถึงการใช้งานหลายๆรีซอร์สหรือจากหลายๆไซต์ได้

ซึ่งรายละเอียดต่างๆของ GSI มีความซับซ้อนมากจึงขอกล่าวถึงในบทถัดไป (บทที่ 4)

3.2.2.2.2. CAS

CAS สร้างขึ้นโดย Globus Toolkit ใช้ใน GSI ซึ่ง CAS เป็นส่วนที่ยอมให้ส่วนที่จัดเตรียมรีซอร์ส ยอมให้มีการสื่อสารโดยควบคุมด้วยนโยบายสำหรับการเข้าถึงมัน ซึ่งจะใช้หลักของดีลิเกชั่น (delegation) ในการควบคุมการจัดการในส่วนนี้



รูปที่ 3-9 แสดงการทำงานของ CAS

การทำงานของ CAS

1. CAS Server จะเริ่มสร้างช่องทางการสื่อสาร โดยจะได้มาซึ่ง GSI ที่เตรียมไว้สำหรับให้ CAS server กำหนดการสื่อสาร
2. ตัวจัดเตรียมรีซอร์สจะได้สิทธิสำหรับการสื่อสารนั้นๆ โดยแต่ละตัวจัดเตรียมนั้นจะเปรียบเสมือนผู้ถือครองการสื่อสารและมีนโยบายการสื่อสารเป็นของตนเอง เมื่อสร้างการเชื่อมต่อที่เชื่อถือได้แล้ว ตัวจัดเตรียมรีซอร์สก็จะยอมให้มีหลักฐานสำหรับการสื่อสารนั้น โดยใช้วิธีการภายในของมัน (เช่น ใช้ กริดแม็พไฟล์และดิสก์โควต้า, การยินยอมจากระบบไฟล์ เป็นต้น)
3. ตัวแทนการสื่อสารนั้นจะใช้ CAS เพื่อจัดการความเชื่อถือได้ของการสื่อสารนั้น และยอมให้ได้รับสิทธิการเข้าถึงรีซอร์สนั้น โดย CAS server จะมีนโยบายของมันที่ใช้ในการจัดการ เช่น สมาชิกของการสื่อสารต้องการสิทธิเพื่อเพิ่มสมาชิกของการสื่อสารนั้น หรือ ขอสิทธิสำหรับเข้าถึงรีซอร์สทั้งหมดของการสื่อสารนั้น
4. เมื่อผู้ใช้งานต้องการที่จะเข้าถึงรีซอร์สได้ร้องขอไปยัง CAS server แล้ว ถ้าฐานข้อมูลของ CAS server ยอมให้ผู้ใช้นั้นใช้งาน CAS จะแจก GSI restricted proxy เพื่อให้ผู้ใช้สามารถทำได้ตามที่ร้องขอไว้
5. เมื่อผู้ใช้ได้รับการเชื่อมต่อกับรีซอร์สของ CAS ด้วยเครื่องมือปกติของ Globus แล้ว (เช่น GridFTP) รีซอร์สที่ผ่านการยอมรับก็จะส่งไปยังผู้ที่ได้รับการยินยอมให้เข้าถึง และได้รับ

สิทธิการเข้าถึงตามที่จำกัดไว้ตามนโยบายของ CAS โดยจะมีข้อจำกัดตามแต่สิทธิของผู้ใช้แต่ละคนซึ่ง CAS ยินยอมให้ตัวจัดเตรียมรีซอร์สสื่อสารด้วย

3.2.2.3. Data Management (การจัดการข้อมูล)

3.2.2.3.1. GridFTP

GridFTP เป็นโปรโตคอลที่ใช้ในการส่งข้อมูลไปบน wide-area networks ที่มีประสิทธิภาพ, ปลอดภัย และเชื่อถือได้ ซึ่ง GridFTP นั้นมีพื้นฐานมาจาก FTP ซึ่งเป็นโปรโตคอลที่นิยมใช้ในการส่งข้อมูลในระบบอินเตอร์เน็ต ซึ่งผู้พัฒนาได้เพิ่มและขยายส่วนต่างๆที่จำเป็นสำหรับการใช้ในระบบกริด

ซึ่งคุณสมบัติที่ GridFTP จัดเตรียมไว้ในระบบกริด มีดังต่อไปนี้

- ทำงานร่วมกับ GSI security เพื่อควบคุมช่องทางข้อมูลให้ปลอดภัย
- ส่งข้อมูลแบบหลายๆทางแบบขนานไปพร้อมๆกัน
- เลือกส่งไฟล์บางส่วน
- ส่งไปยังกลุ่มที่ 3 (ส่งตรงจากเซิร์ฟเวอร์ไปยังเซิร์ฟเวอร์)
- รับรองช่องทางข้อมูล
- ใช้ช่องทางข้อมูลใหม่
- ทำไปป์ไลน์ของคำสั่งได้

3.2.2.3.2. RLS

Replica Location Service (RLS) เป็นส่วนที่ดูแลและจัดเตรียมการเข้าสู่การแม็พจากข้อมูลที่เป็นลอจิคให้เป็นค่าข้อมูลที่จะใช้ที่เป้าหมาย ซึ่งส่วนนี้ได้พัฒนาโดยทีมที่ทำเรื่อง DataGrid ซึ่งจะช่วยลดเวลาในการเข้าถึงข้อมูล เพิ่มการวางตำแหน่งที่ตั้งของข้อมูล และเพิ่มความแข็งแรง, ประสิทธิภาพของแอพพลิเคชั่นแบบกระจาย

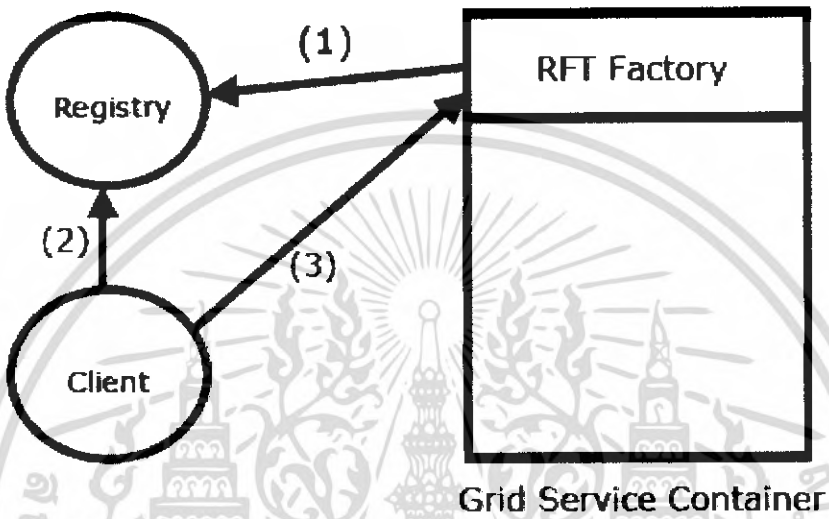
RLS นั้นจะมีหน้าที่ในด้านต่อไปนี้

- Consistent local state maintained in Local Replica Catalogs (LRCs): จะใช้ดูแลการแม็พระหว่าง arbitrary logical file names (LFNs) และ psysical file names (PFNs) โดยเกี่ยวข้องกับระบบของอุปกรณ์การเก็บข้อมูล
- Collective state with relaxed consistency mainained in Replica Location Indices (RLIs): แต่ละ RLI จะประกอบไปด้วยกลุ่มของการแม็พจาก LFNs เป็น LRCs
- Soft state maintenance of RLI state: LRC จะส่งข้อมูลที่เกี่ยวกับสเตทของ RLIs โดยใช้ซอฟท์สเตทโปรโตคอล ซึ่งค่าข้อมูลสเตทนั้นจะต้องรีเฟรทอย่างสม่ำเสมอ
- Compression of state update: ใช้รวมเนื้อหาทั้งหมดของ Local Replica Catalog ก่อนที่จะส่งไปยังการอัพเดตแบบซอฟท์สเตทที่ Replica Location Index Node

- Membership and partitioning information maintenance: RLS จะใช้ดูแลข้อมูลที่เป็นแบบ static ที่เกี่ยวกับ LRCs และ RLIs ในระบบแบบกระจาย ซึ่งจะใช้กลไกของ OGSA สำหรับการลงทะเบียนเซอร์วิสและใช้จัดการช่วงเวลาชีวิตของเซอร์วิสนั้น

3.2.2.3.3. RFT

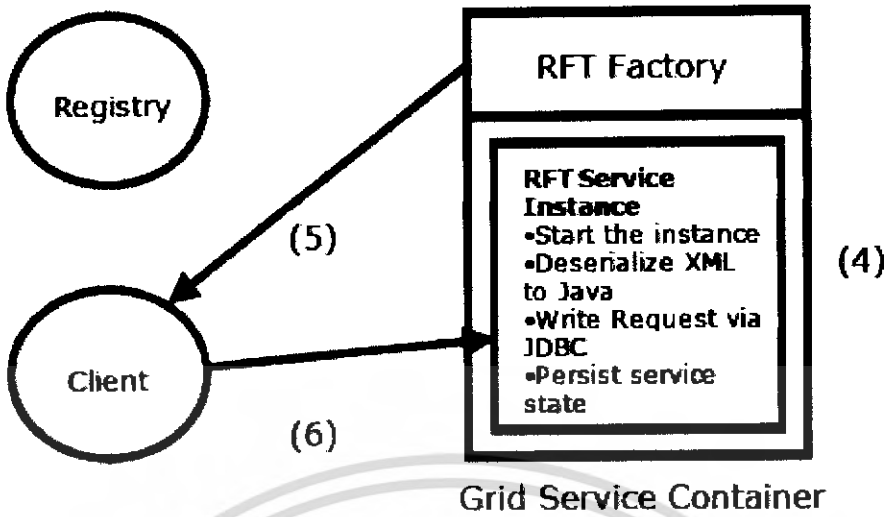
Reliable File Tranfer (RFT) เป็นเซอร์วิสพื้นฐานของ OGSA ที่ใช้สำหรับควบคุมดูแลการถ่ายโอนข้อมูลระหว่างผู้ใช้งานกลุ่มที่ 3 ที่ใช้ GridFTP ซึ่งมีหลักการทำงานดังภาพ



รูปที่ 3-10 แสดงขั้นตอนการทำงานของ RFT

การทำงานเริ่มเมื่อกริดเซอร์วิสคอนเทนเนอร์เริ่มทำงาน โดยบรรจุ RFT factory ไว้

1. RFT Factory นั้นจะเริ่มลงทะเบียนตัวของมันให้เป็นค่ารีจิสตรีที่เป็นที่รู้จักกัน
2. ไคล์เอนท์ค้นเจอ RFT Factory นั้นโดยการคิวรี่ serviceData ของรีจิสตรีนั้น
3. ไคล์เอนท์เรียกคำสั่ง createService บนแฟคทอรี่นั้นและส่งไปยังที่ที่ร้องขอ



รูปที่ 3-11 แสดงขั้นตอนการทำงานของ RFT (ต่อ)

4. เซอร์วิสอินสแตนซ์นั้นเริ่มทำงาน

5. RFT factory จะส่งค่า locator กลับไปยังไคล์เอนท์

6. ไคล์เอนท์ จะเรียก Start() และส่งค่าไปยังส่วน notifications หรืออื่นๆ

3.2.2.3.4. XIO

Globus XIO เป็นส่วนที่ใช้สำหรับขยายอินพุท/เอาท์พุท ไลบรารี่สำหรับ Globus Toolkit ที่ใช้จัดเตรียม API พื้นฐาน (เช่น open/close/read/write) เพื่อนำไปใช้ใน IO ตามต้องการ ซึ่ง Globus XIO นั้นมีจุดหมายสำคัญอยู่ 2 ประการ คือ

- เตรียม API สำหรับผู้ใช้แบบเดี่ยวให้กับโปรโตคอลที่เกี่ยวข้องกับ IO ของกริด ส่วนนี้จะใช้การส่งข้อความสำหรับใช้งานในโปรโตคอลที่แตกต่างกัน ซึ่งผู้เขียนโปรแกรมจำเป็นต้องเขียนให้อยู่ในรูปแบบที่กำหนดไว้ใน API นั้นๆ ซึ่งโปรโตคอลที่สร้างให้ Globus XIO นั้นจะเรียกว่า "drivers" ซึ่งเมื่อใช้งานจะเป็นส่วนที่ซ่อนจากผู้พัฒนา โดยจำเป็นต้องเขียน API นี้ในครั้งแรก เมื่อจะใช้แอพพลิเคชั่นนั้นๆในครั้งต่อๆไป จะสามารถทำได้โดยไม่ต้องเปลี่ยนซอร์สโค้ดเลย
- ลดเวลาในการสร้างหรือทดลองโปรโตคอลใหม่ๆ โดย Globus XIO นั้นจะจัดเตรียม "driver" ดังที่กล่าวไปแล้ว ทำให้ไม่จำเป็นต้องยุ่งมากนักในการใช้โปรโตคอลใหม่ๆ และยังเป็นส่วนที่ทำให้เกิดการนำซอร์สโค้ดเก่ามาใช้งานใหม่ด้วย และยังสามารถนำไปรวมกับ driver อื่นๆได้

Globus XIO นั้นมีฟีเจอร์ที่เกี่ยวข้องกับระบบการทำงาน โดยได้เตรียม API ที่ใช้งานได้ให้กับผู้ใช้งาน ทำให้ผู้ใช้งานไม่ต้องเขียนฟังก์ชั่นยากๆด้วยตัวเองอย่างเดียว โดยฟีเจอร์ของ XIO นั้น มีดังนี้

- Simple API ใช้สำหรับงานพื้นฐาน เช่น open, close, read, write
- Extensibility with drivers เนื่องจาก Globus XIO นั้นออกแบบให้ขยายได้ง่าย ทำให้สามารถเพิ่มฟีเจอร์ใหม่ๆ เข้าไปได้
- Effeciency เนื่องจาก Globus XIO ได้ถูกออกแบบให้มีประสิทธิภาพ มีไลบรารี่ที่สามารถใช้งานได้หลากหลาย
- Timeouts ทำให้ผู้ใช้งานไม่ต้องกำหนด timeout เอง เพราะ globus xio จะมีค่า timeout โดยปกติของมันอยู่แล้ว และสามารถปิดตัวเองเมื่อทราบว่าคำสั่งนั้นไม่ต้องทำงานแล้วด้วย
- Data Descriptions มีบัฟเฟอร์ที่ให้ผู้ใช้สามารถเขียนหรืออ่านได้ โดย Globus XIO ได้ยอมให้ผู้ใช้ผ่านเข้าไปจัดการส่วนนี้ได้ แม้ว่า IO นั้นจะทำงานสมบูรณ์ไปแล้ว

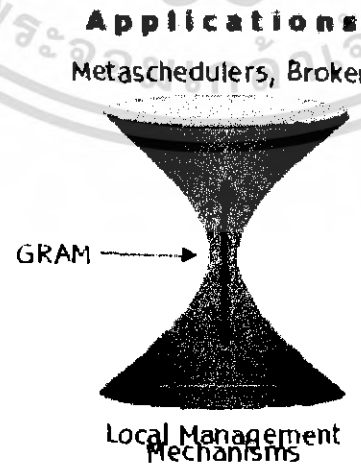
3.2.2.4. Resource Management (การจัดการทรัพยากร)

3.2.2.4.1 GRAM

Globus Toolkit ได้ใช้ *GRAM (Grid Resource Allocation and Management)* ในการจัดการระบบในระยะไกลโดยเตรียมรูปแบบสำหรับการร้องขอและใช้ระบบจากระยะไกลสำหรับการทำงาน โดยส่วนที่ใช้มากที่สุดก็คือ ใช้สำหรับจ่ายและควบคุมงานจากระยะไกล

GRAM ได้ออกแบบโดยใช้โปรโตคอลพื้นฐานและ API สำหรับการร้องขอและใช้รีซอร์สจากระบบในระยะไกล โดยอาศัยรูปแบบที่อ้างจากตารางงาน (Job Schedule) โดยอาศัย GSI มาช่วยเสริมในด้านความปลอดภัย

GRAM จะใช้หลากหลายวิธีในการควบคุมงาน คือใช้ ตารางงาน, ระบบคิวรี่, ระบบสำรอง และรูปแบบการควบคุม ซึ่งทำให้ผู้พัฒนาสามารถเรียนรู้มันและนำไปช่วยในการพัฒนาระบบได้ ซึ่งถ้าเปรียบแล้ว GRAM จะทำงานเหมือนกับคอของนาฬิกาทรายที่ใช้แอพพลิเคชั่นและเซอร์วิสระดับสูง (เช่น resource broker) มาช่วยในการให้สิทธิการเข้าถึงนั่นเอง



รูปที่ 3-12 ลักษณะการทำงานของ GRAM

3.2.2.5. Information Service (การให้บริการข้อมูล)

3.2.2.5.1. MDS3

Monitoring and Discovery Services (MDS3) เป็นส่วนที่จัดการเกี่ยวกับข้อมูลเซอร์วิส สำหรับ Globus Toolkit 3 ซึ่งจะใช้จัดการกับพวก serviceData และ Index Service โดยมีวิธีการคิวรี่ผ่าน XML รูปแบบหนึ่งที่เรียกว่า XPath

serviceData ที่ใช้ใน GT3 ก็คือ ข้อมูล XML ที่ถูกสร้างโดยทุกๆกริดเซอร์วิสที่แทนในสเตทภายใน ซึ่งแต่ละส่วนของ serviceData นั้นจะถูกเรียกว่า serviceData Element (SDE) ดังที่กล่าวไปแล้วในบท OGSI ซึ่งสเตทของโฮสท์นั้นๆจะถูกแสดงโดย SDE แบบเดี่ยว (ด้วย GRAM) หรือจากสถานะของงาน

วิธีการคิวรี่แบบ XPath ที่ใช้ใน MDS นั้น สร้างขึ้นมาจาก XATAN XPath library ซึ่งเป็นส่วนหนึ่งของภาษาคิวรี่แบบ W3C XML โดยที่เซอร์วิสใดๆที่ถูกสร้างขึ้นจาก GT3 OGSA แล้วจะต้องรองรับการทำงานของฟังก์ชั่นนี้ โดย Globus ได้กำหนดรูปแบบของการคิวรี่อินพุท โดยใช้ค่า SDE และ XPath โดยที่เอาท์พุทที่ได้เป็นผลลัพธ์จากการคำนวณของ Xpath คิวรี่นั้นกับกลุ่มของ SDE ที่พิจารณา

บทที่ 4

Grid Security Infrastructure

(โครงสร้างของความปลอดภัยในระบบกริด)

4.1.ความรู้พื้นฐาน

4.1.1. การรักษาความปลอดภัยในการติดต่อสื่อสาร

การติดต่อสื่อสารกันอย่างมีความปลอดภัยนั้นไม่ได้หมายความว่ามีเพียงการเข้ารหัสและถอดรหัสข้อมูลเท่านั้นที่สามารถทำให้การสื่อสารนั้นปลอดภัยได้ แต่ยังมีในส่วนอื่นๆด้วยอันได้แก่

- การรักษาความลับ(privacy) ในการสื่อสารกันนั้น จำเป็นที่จะต้องเป็นความลับ คือไม่มีผู้อื่นรู้ความใดๆในการส่งนั้นๆ เว้นแต่ผู้ที่รับและส่งข้อมูลเท่านั้นที่รู้ความหมายและหากข้อมูลที่ส่งถูกขโมยไประหว่างทาง ผู้ที่ขโมยไปนั้นจะไม่สามารถรู้ความหมายได้ ซึ่งกลไกที่ทำหน้าที่รักษาความลับคือการเข้ารหัส และ ถอดรหัสนั่นเอง
- การรักษาความถูกต้อง(integrity) เมื่อมีการรับและส่งข้อมูลการรักษาความถูกต้อง จะทำหน้าที่คือทำให้ผู้รับข้อมูลแน่ใจได้ว่าข้อมูลที่ได้รับมาถูกต้อง และไม่ได้รับการเปลี่ยนแปลงไประหว่างทางที่ทำการส่ง
- การพิสูจน์ตัวตน(authentication) ทำหน้าที่คือป้องกันการปลอมแปลงเป็นบุคคลใดบุคคลหนึ่ง ป้องกันการแอบอ้าง ซึ่งถ้าทำในส่วนของการพิสูจน์ตัวตน เรียบร้อยแล้วจะมีกลไกถัดไปที่เข้ามาช่วยในการทำงานนี้คือ การพิสูจน์สิทธิ์ (authorization) เป็นกลไกที่ช่วยในการกำหนดความสามารถของผู้ใช้ระบบนั้นในการทำงานกับระบบ

ซึ่งทั้ง 3 ส่วนดังได้กล่าวมานั้น ในบางระบบอาจมีความจำเป็นไม่เท่ากัน เช่น ต้องการให้มีส่วนของการรักษาความถูกต้องมากกว่า แต่ไม่มีความจำเป็นต้องให้มีการการพิสูจน์ตนก็ได้

ระบบความปลอดภัยถือเป็นพื้นฐานที่มีความสำคัญในการออกแบบระบบกริด ซึ่งใน globus toolkit จะมีระบบรักษาความปลอดภัยเบื้องต้น ดังได้กล่าวมาแล้ว หากไม่มีฟังก์ชั่นที่ช่วยรักษาความปลอดภัยเหล่านี้แล้ว ระบบกริดจะมีความเสี่ยงในการเสียหายสูง ซึ่งใน Globus toolkit จะมี grid security Infrastructure และ public key infrastructure เป็นส่วนที่ช่วยจัดการในเรื่องของความปลอดภัยในส่วนนี้

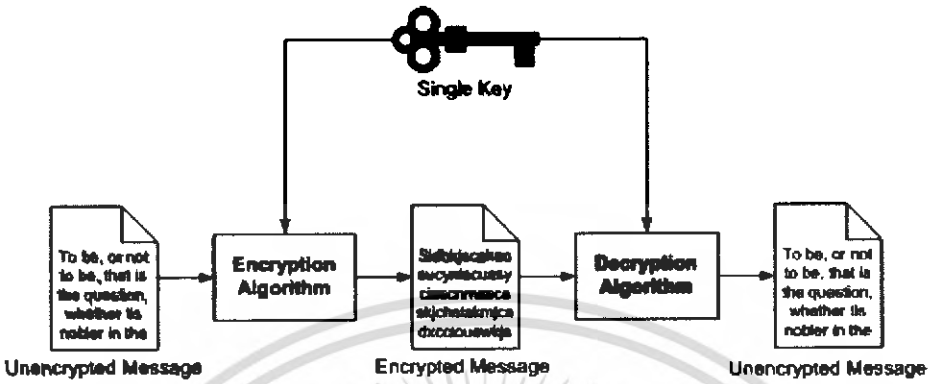
4.1.2. การเข้ารหัสเบื้องต้น

4.1.2.1. การเข้ารหัสโดยการใช้คีย์

ในอดีตการเข้ารหัสและถอดรหัสนั้นไม่ได้ใช้คีย์ดังในปัจจุบัน แต่ใช้การแทนที่ของตัวอักษรแทน แต่เกิดปัญหาขึ้นคือสามารถถอดรหัสได้ง่ายทำให้การเข้ารหัสที่ต้องการรักษาข้อมูลให้เป็นความลับไม่เป็นความลับอีกต่อไปจึงเปลี่ยนมาใช้คีย์แทน

4.1.2.2. การเข้ารหัสแบบคีย์สมมาตร

การเข้ารหัสแบบคีย์สมมาตรซึ่งใช้เพื่อเข้ารหัสและถอดรหัส เพื่อให้เกิดความมั่นใจว่าข้อมูลที่ทำการส่ง จะมีแค่ผู้รับกับผู้ส่งเท่านั้นที่สามารถอ่านได้ สิ่งที่สำคัญที่สุดคือจะต้องรักษาคีย์ให้ปลอดภัย และเป็นความลับไม่ให้ผู้อื่นรู้ และถ้ามีบุคคลที่สามล่วงรู้คีย์จะสามารถเข้ารหัสและถอดรหัสได้ทันที



รูปที่ 4-1 การเข้ารหัสแบบคีย์สมมาตร

ตัวอย่างของการเข้ารหัสแบบนี้คือ

- Data Encryption Standard (DES): ใช้คีย์ทั้งหมด 56 บิท บวกกับ parity bit จำนวน 8 บิท

Triple-DES: ใช้คีย์ขนาด 112 บิท บวกกับ parity bit จำนวน 16 บิท หรือ คีย์ขนาด 168 บิท บวกกับ parity bit จำนวน 24 บิท

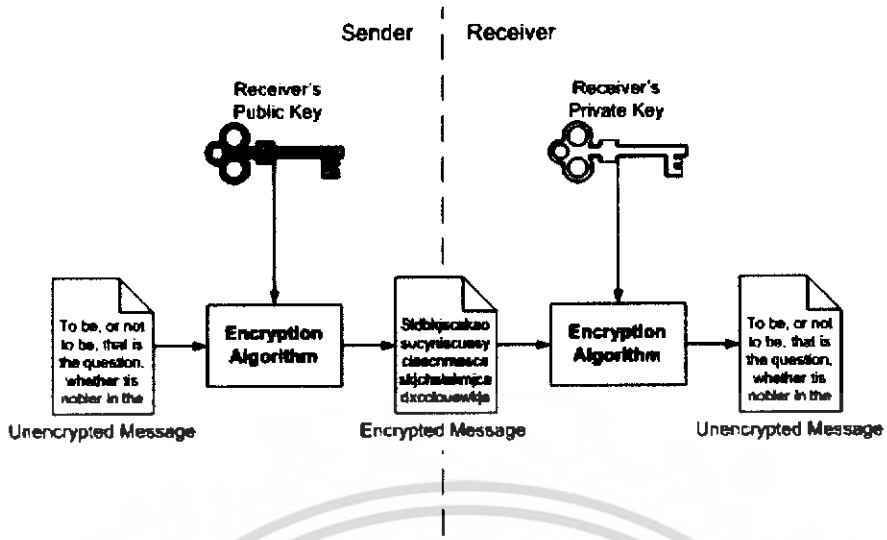
- RC2 และ RC4: เป็นการเข้ารหัสที่คีย์เปลี่ยนแปลงขนาดได้ มักจะใช้ 40 บิท หรือ 128 บิท

การเข้ารหัสและถอดรหัส โดยการใช้คีย์แบบสมมาตรเป็นวิธีการที่ง่ายและมีความรวดเร็วของกระบวนการเข้ารหัส แต่มีความปลอดภัยต่ำ ปัจจุบันจึงมีแนวโน้มในความนิยมต่ำเช่นกัน

4.1.2.3. การเข้ารหัสแบบคีย์ไม่สมมาตร

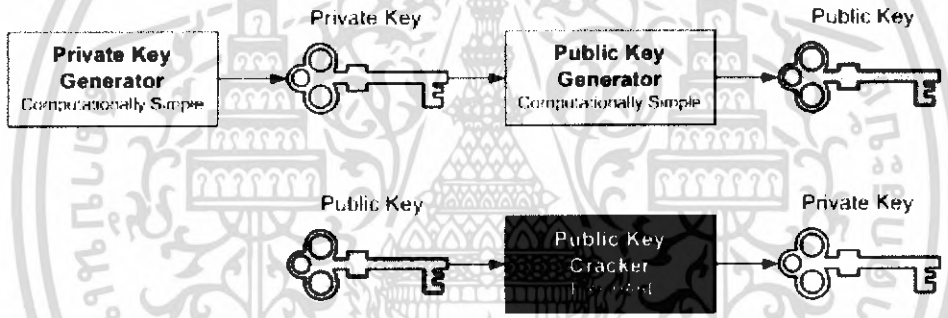
เป็นวิธีการที่ได้รับความนิยมมาก หรืออาจจะเรียกได้ว่า public key cryptography (RSA) ในการเข้ารหัสแบบนี้จะมีคีย์ 2 คีย์ที่ไม่เหมือนกัน ซึ่งจะเรียกว่า คีย์สาธารณะ(public key) และ คีย์ส่วนตัว (private key) โดยใช้ปกติจะใช้คีย์ส่วนตัวในการเข้ารหัสและคีย์สาธารณะในการถอดรหัส ซึ่งสิ่งที่ต้องรักษาให้มีความปลอดภัยมากก็คือคีย์ส่วนตัวซึ่งโดยทั่วไป คีย์สาธารณะอาจจะให้ผู้อื่นรู้ได้ และมักจะอยู่ในรูปแบบของ ดิจิตอลซิกเนเจอร์ (digital signature) ซึ่งเป็นส่วนหนึ่งของผู้ประกอบการรับรอง (certificate Authority ,CA)

กระบวนการเข้ารหัสและถอดรหัสแบบคีย์ไม่สมมาตร คือต้องการให้ข้อมูลที่ทำการเข้ารหัสสามารถถอดรหัสด้วยคีย์ที่เป็นคู่ของกันเท่านั้น ซึ่งเราสามารถใช้วิธีการทางคณิตศาสตร์ช่วยเพิ่มประสิทธิภาพให้การเข้ารหัสและถอดรหัสมีประสิทธิภาพมากขึ้น แต่ต้องการเวลาในการเข้ารหัสมากขึ้น และการเข้ารหัสแบบคีย์ไม่สมมาตรนั้นมีประสทธิภาพมากขึ้นกว่าการเข้ารหัสแบบคีย์สมมาตร



รูปที่ 4-2 แสดงการเข้ารหัสแบบคีย์ไม่สมมาตร

ข้อดีของการเข้ารหัสแบบคีย์ไม่สมมาตร

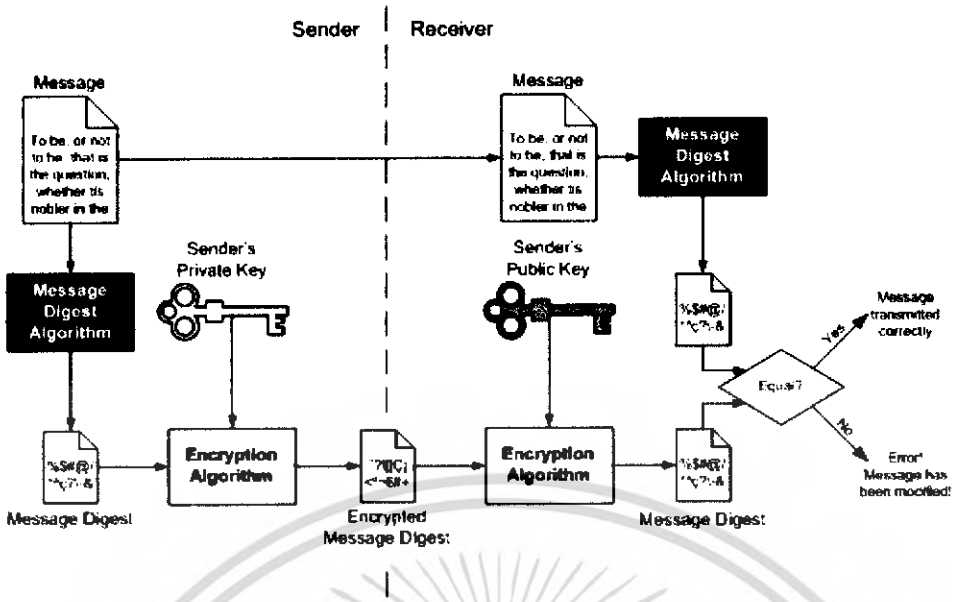


รูปที่ 4-3 แสดงการทำลายคีย์

- เมื่อเปรียบเทียบกับการเข้ารหัสแบบคีย์สมมาตร ไม่มีความจำเป็นต้องตกลงคีย์ด้วยกันทั้งสองฝ่าย แค่เพียงอีกฝ่ายมีคีย์สาธารณะเท่านั้นก็เพียงพอแล้ว
- ความสัมพันธ์ระหว่างคีย์สาธารณะและคีย์ส่วนตัวคือเป็นกลไกที่ง่ายในการคำนวณคีย์สาธารณะจากคีย์ส่วนตัวแต่เป็นกลไกที่ยากหากต้องการคำนวณหาคีย์ส่วนตัวจากคีย์สาธารณะ ดังรูปภาพที่ 4-3

4.1.3. ดิจิตอลซิกเนเจอร์ (Digital signatures)

กลไกการทำการรักษาความถูกต้อง ในระบบโครงสร้างของการเข้ารหัสแบบไม่สมมาตรจะใช้ดิจิตอลซิกเนเจอร์ซึ่งเป็นส่วนหนึ่งของข้อมูลที่แนบไปกับข้อมูลจริงๆ เป็นส่วนที่ทำให้สามารถพบได้ว่าข้อมูลจริงที่ทำการส่งไปนั้น เป็นข้อมูลที่ถูกต้องโดยที่ระหว่างทางที่ทำการส่งไม่มีผู้ใดที่สามารถเปลี่ยนแปลงข้อมูลนั้นได้ มีขั้นตอนการทำงานดังต่อไปนี้



รูปที่ 4-4 กลไกการรักษาความถูกต้องในกระบวนการเข้ารหัสด้วยโครงสร้างของคีย์สาธารณะ

- สร้างแมสเซสไดเจสท์ซึ่งเป็นผลของข้อมูลที่ต้องทำการส่ง มีคุณสมบัติคือมีขนาดสั้นกว่าข้อมูลที่ทำการส่งและเป็นข้อมูลที่ผ่านกลไกการคำนวณ และเมื่อข้อมูลเปลี่ยนไปเพียงเล็กน้อยความแตกต่างของ แมสเซสไดเจสท์สองชิ้นนั้นมีความแตกต่างกันอย่างสิ้นเชิง
- แมสเซสไดเจสท์ จะได้รับการเข้ารหัสด้วยคีย์ส่วนตัว และผลของแมสเซสไดเจสท์ นั้นเรียกว่า ดิจิตอลซิกเนเจอร์

เมื่อ ดิจิตอลซิกเนเจอร์ได้รับการแนบไปกับข้อมูลที่ต้องการส่ง และถูกส่งไปยังผู้รับที่ต้องการแล้วผู้รับจะปฏิบัติตามขั้นตอนดังต่อไปนี้

- ใช้คีย์สาธารณะของผู้ส่งทำการถอดรหัสดิจิตอลซิกเนเจอร์
- ใช้กลไกของการสร้าง แมสเซสไดเจสท์แบบเดียวกับที่ผู้ส่งใช้ เพื่อหาแมสเซสไดเจสท์จากข้อมูลที่ได้รับ
- เปรียบเทียบแมสเซสไดเจสท์ ของผู้ส่งและของตนเองที่เพิ่งสร้างขึ้นมาดังข้อข้างต้นว่าเหมือนกันหรือไม่

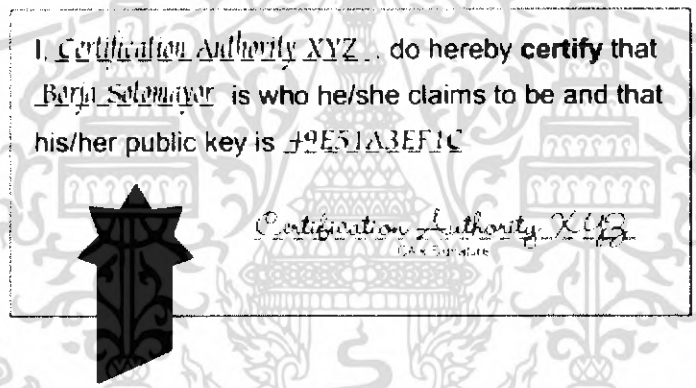
4.1.3.1. ใบรับรองดิจิตอล (Digital certificate)

ใบรับรองดิจิตอลเป็นเอกสารทางดิจิตอล ที่เกี่ยวเนื่องกับทรัพยากรในระบบกริดและ คีย์สาธารณะใบรับรอง(certificate) คือโครงสร้างของข้อมูลที่บรรจุคีย์สาธารณะและข้อมูลที่ตรงกับเจ้าของคีย์ ซึ่งเอกสารฉบับนี้จะได้รับการ ไซน์ (sign) จากกลุ่มหรือองค์กรอื่นๆที่เรียกว่าผู้ให้บริการออกใบรับรอง (CA) ใบรับรองดิจิตอลในกริดนั้นใช้มาตรฐานของ X.509 certificate ซึ่งอาจเปรียบได้กับ

หนังสือเดินทาง แต่แตกต่างกันตรงที่ใบรับรองนี้ทำหน้าที่ในการแยกแยะทรัพยากรในระบบกริด และยังสามารถแจกจ่ายสำเนาของใบรับรองนี้ได้อย่างไม่มีเงื่อนไข ใบรับรองนี้ไม่ได้บรรจุข้อมูลที่มีความสำคัญและต้องรักษาความลับไว้ สามารถแจกจ่ายให้ผู้อื่นได้โดยไม่เกิดความเสี่ยงในด้านความปลอดภัย

เป็นความจริงที่ว่าใบรับรองดิจิตอลแนบไปกับคีย์สาธารณะ ของผู้ที่มีรายนามในใบรับรองนั้น การสร้างใบรับรองดิจิตอล มีโอกาสที่จะมีผู้เข้ามาเปลี่ยนแปลงข้อมูลในใบรับรองน้อยมากเพราสามารถตรวจพบได้ทันทีด้วยกลไกการตรวจสอบความถูกต้องของข้อมูล

เมื่อเครื่องไคล์เอนท์ที่อยู่ในระบบกริดต้องการที่จะติดต่อไปยังจุดหมายปลายทางใดๆภายในระบบ จะไม่มีความจำเป็นที่จะต้องใช้คีย์สาธารณะจะใช้เพียงใบรับรองเท่านั้นและเมื่อจุดหมายปลาย ทางได้รับการติดต่อจากเครื่องไคล์เอนท์เครื่องหนึ่งในระบบแล้วจุดหมายปลายทางจะตรวจสอบ ดิจิตอลซิกเนเจอร์ของผู้ให้บริการออกใบรับรองที่มีอยู่ใน ใบรับรองและหากพบว่าถูกต้องจะได้รับความเชื่อถือจุดหมายปลายทางจึงจะสามารถรับคีย์สาธารณะ จากเครื่องไคล์เอนท์ที่เข้ามาทำการติดต่อไว้ได้ จึงจะรับประกันได้ว่าไม่มีใครปลอมแปลงคีย์สาธารณะในระบบ



รูปที่ 4-5 ใบรับรอง

เปรียบเทียบเหตุการณ์ได้จากรูปภาพที่ 4-5 คือ ใบรับรองนั้นได้รับการ ไซน์จากองค์กรอื่น และ CA's signature นั้นคือ ดิจิตอลซิกเนเจอร์ที่ได้รับการสร้างขึ้นจากคีย์ส่วนตัวของผู้ให้บริการออกใบรับรองเท่านั้น ในขณะเดียวกันสามารถตรวจสอบการรักษาความถูกต้องได้โดยการใช้คีย์สาธารณะของผู้ให้บริการออกใบรับรอง ใบรับรองที่ใช้คือใช้ตามมาตรฐานของ X.509 ซึ่งมีรูปแบบดังต่อไปนี้

X.509 certificate นั้นเป็นเทกซ์ไฟล์แบบหนึ่งที่ประกอบไปด้วยข้อมูลที่อยู่ในรูปแบบของซินแทกซ์เป็นจำนวนมาก ซึ่งซินแทกส์เหล่านี้ทำหน้าที่กำหนดขอบเขตของใบรับรองนั่นเอง องค์ประกอบหลักของ X.509 ใบรับรองมีดังนี้

- หัวเรื่อง(Subject) หมายถึงชื่อของผู้ใช้ ซึ่งจะเข้ารหัสให้อยู่ในชื่อที่เรียกว่า distinguished name ตัวอย่างเช่นมีผู้เข้าใช้ระบบชื่อ Mike Kendzierski จะมี distinguished name ดังนี้ **"/O=Grid/O=GridTest/OU=test.domain.com/CN=Mike Kendzierski"**

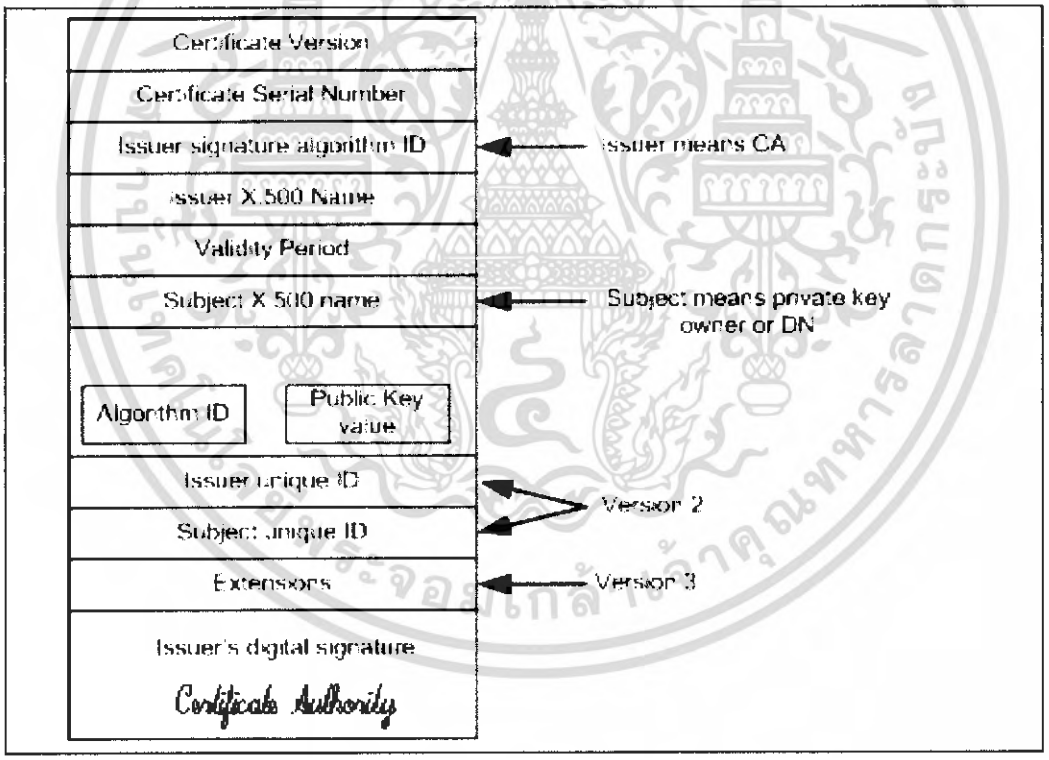
O = Organization

OU = Organizational Unit

CN = Common name (ชื่อของผู้ใช้ระบบ)

C = Country

- คีย์สาธารณะของผู้ใช้ระบบ(Subject's public key) ประกอบไปด้วยคีย์สาธารณะและวิธีการในการสร้างคีย์สาธารณะด้วย
- Issuer's Subject หมายถึง CA's distinguished name หรืออาจเรียกว่าเป็นชื่อของผู้ให้บริการออกใบรับรอง
- ใบรับรองดิจิตอล เป็นใบรับรองที่ประกอบไปด้วย ดิจิตอลซิกเนเจอร์ของข้อมูลทั้งหมดในใบรับรอง โดยผู้ให้บริการออกใบรับรองจะใช้คีย์ส่วนตัว ในการสร้างดิจิตอลซิกเนเจอร์และในการพิสูจน์ว่า ใบรับรองนั้นถูกต้องหรือไม่จะต้องขอคีย์สาธารณะจากผู้ให้บริการออกใบรับรอง และใบรับรองดิจิตอลยังทำหน้าที่ป้องกันไม่ ให้ Distinguished Name – DN ในใบรับรองซ้ำกันอีกด้วย และอาจจะทำหน้าที่เก็บข้อมูลอื่นๆ เช่น ที่อยู่ e-mail , ที่อยู่ขององค์กร



รูปที่ 4-6 ใบรับรองดิจิตอล

การได้รับใบรับรองจากผู้ให้บริการออกใบรับรอง จะต้องดำเนินการดังขั้นตอนต่อไปนี้

1. ผู้เข้าใช้ระบบกริดจะต้องการการรับรองโดยการสร้างคีย์สองคีย์คือ คีย์สาธารณะ , คีย์ส่วนตัว
2. ผู้เข้าใช้ระบบกริดจะต้องเข้ารหัสข้อมูลคือคีย์สาธารณะและข้อมูลต่างๆที่ผู้ให้บริการออกใบรับรองต้องการ

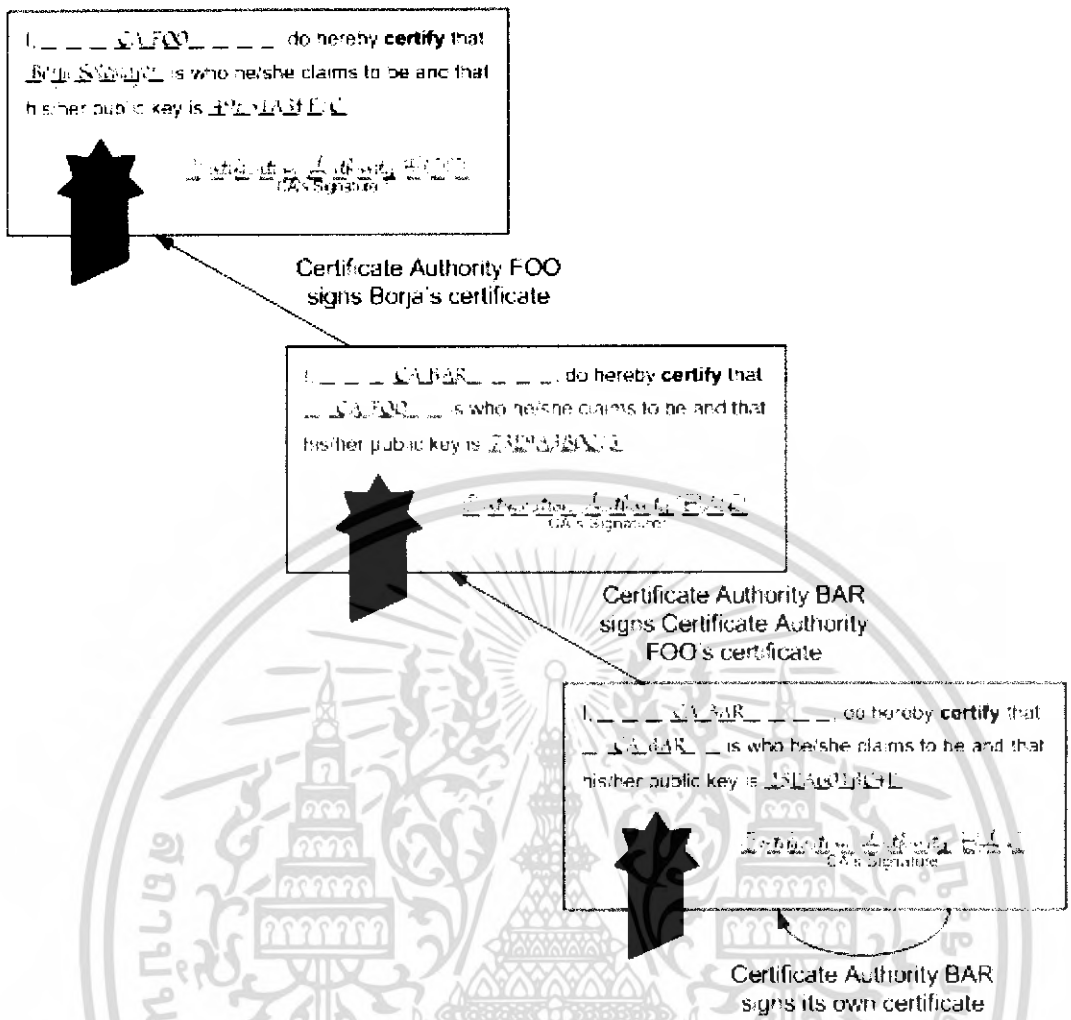
3. ข้อมูลที่ได้รับการเข้ารหัสเรียบร้อยแล้ว จะถูกส่งไปยังผู้ให้บริการออกใบรับรองซึ่งสำหรับคีย์ส่วนตัวนั้นยังถูกเก็บเป็น ความลับอยู่ทางฝั่งเครื่องไคล์เอนท์ไม่ได้ส่งไปยังผู้ให้บริการออกใบรับรองด้วย
4. ผู้ให้บริการออกใบรับรองจะทำการพิสูจน์ว่าผู้ใช้ระบบกริดผู้นั้นเป็นเจ้าของคีย์สาธารณะจริง
5. ผู้ให้บริการออกใบรับรองต้องการที่จะพิสูจน์ความเป็นผู้ใช้ระบบกริด ผู้ให้บริการออกใบรับรองสามารถใช้ข้อมูลที่บันทึกไว้ในระบบเพื่อตรวจสอบตัวตนของผู้ใช้ระบบ

เมื่อผู้ให้บริการออกใบรับรองทำการตรวจสอบแล้วว่าข้อมูลทั้งหมดถูกต้อง ผู้ให้บริการออกใบรับรองจะทำการสร้างใบรับรองด้วยการไซน์เข้าใช้ระบบ ด้วยคีย์สาธารณะของผู้ใช้ระบบกริดคนนั้น และใบรับรองนี้จะถูกส่งไปยังผู้ใช้ระบบทุกคนในขณะนั้น

4.1.3.2. ผู้ให้บริการออกใบรับรอง (Certification Authority)

- กระบวนการในการออกใบรับรองดิจิตอลให้กับผู้ที่ทำการร้องขอมายังผู้ให้บริการออกใบรับรอง ผู้ขอใบรับรองสร้างคู่ของคีย์และส่งเฉพาะคีย์สาธารณะ มาพร้อมกับข้อมูลส่วนบุคคลที่อยู่ในรูปของCertificate Request(คำขอใบรับรอง) ไปยังผู้ให้บริการออกใบรับรองซึ่งจะต้องผ่านกระบวนการตรวจสอบตัวบุคคลอย่างละเอียดโดยหน่วยงานรับลงทะเบียน หลังจากนั้นคำขอใบรับรองที่ผ่านการตรวจสอบจะถูกนำไปสร้างเป็นใบรับรองโดย ผู้ให้บริการออกใบรับรองและถูกส่งกลับมายังผู้ขอเพื่อนำไปใช้งานต่อไปโดยใบรับรองที่ได้จะถูกทำสำเนาไว้ที่ Directory Service(ระบบบริการไดเรกทอรี) อีกชุดหนึ่ง เพื่อให้บุคคลทั่วไปสามารถค้นหาได้ สะดวกสำหรับอัลกอริธึมที่ใช้ในการสร้าง คีย์แพร์ (Key pair) และขนาดของคีย์แพร์จะถูกกำหนดโดย ผู้ให้บริการออกใบรับรองโดย อัลกอริธึมที่นิยมใช้คือ RSA(Ron Rivest, Adi Shamir and Leonard Adleman), DSA (Digital Signature Algorithm) ที่มีขนาด 512 ถึง 1,024 บิตหากมีจำนวนบิตสูงขึ้นจะยิ่งมีความปลอดภัยมากขึ้น แต่จะส่งผลให้การเข้ารหัส/ถอดรหัสข้อมูลใช้เวลามากขึ้น ซึ่งความปลอดภัยจะขึ้นอยู่กับความยาวของคีย์แพร์สำหรับ คีย์แพร์ของผู้ให้บริการออกใบรับรองที่นิยมใช้จะถูกสร้างด้วยอัลกอริธึม RSA ที่มีขนาด 2,048 บิตขึ้นไป

  ใน globus toolkit จะมี simple CA ที่สามารถทำหน้าที่นี้ได้
- รูปแบบการมอบความไว้วางใจของผู้ให้บริการออกใบรับรอง



รูปที่ 4-7 รูปแบบกระบวนการมอบความไว้วางใจเป็นลำดับชั้น

รูปแบบของ Trust Model (การมอบความไว้วางใจ) ของ ผู้ให้บริการออกใบรับรอง จะเป็นแบบลำดับชั้นที่แน่นอน(Strict Certification Hierarchy) กล่าวคือใบรับรองของบุคคลใดๆ จะถูกสร้างโดยผู้ให้บริการออกใบรับรองเท่านั้น โดยผู้ให้บริการออกใบรับรองสามารถมีได้หลายลำดับชั้น ชั้นบนสุดจะเรียกว่า ผู้ให้บริการออกใบรับรองหลัก(Root CA) โดยที่ ใบรับรองของผู้ให้บริการออกใบรับรองหลักจะเป็น เซลฟ์ไซน์ (Self-signed) และจะเป็นผู้ออกใบรับรองสำหรับผู้ให้บริการออกใบรับรองในชั้นถัดลงมา(Intermediate CA) ที่อยู่ติดกันเป็นลำดับไปเรื่อยๆ จนไปสิ้นสุดที่การออกใบรับรองสำหรับ ผู้ใช้งานการมีผู้ให้บริการออกใบรับรองแบบหลายลำดับชั้นดังจะเห็นจากรูปภาพที่ 4-7 มีประโยชน์ในแง่ของการแบ่งกลุ่มผู้ใช้งานออกเป็นหลายๆ กลุ่มเนื่องจากจำนวนผู้ใช้งานมาก ดังนั้นจึงมีการแบ่งกลุ่มผู้ใช้ตาม หน่วยงานของผู้ใช้ ตามความสามารถในการใช้งานใบรับรองหรือตามระดับความรับผิดชอบของผู้ให้บริการออกใบรับรอง เป็นต้น

4.1.3.2.1.หน้าที่ของผู้ให้บริการออกใบรับรอง

1. ผู้ให้บริการออกใบรับรอง จะสุ่มค่าเพื่อสร้างคีย์สองคีย์ที่เป็นคู่ของกัน
2. ผู้ให้บริการออกใบรับรองทำหน้าที่รักษาคีย์ส่วนตัวของผู้ให้บริการออกใบรับรองไม่ให้ผู้อื่นได้รับรู้
3. ผู้ให้บริการออกใบรับรองสร้างใบรับรองที่เป็นของตัวเอง
4. ผู้ให้บริการออกใบรับรองทำการเข้ารหัสใบรับรองที่สร้างขึ้นด้วย คีย์ส่วนตัวของผู้ให้บริการออกใบรับรอง
6. คีย์ส่วนตัวของผู้ให้บริการออกใบรับรอง เป็นส่วนที่สำคัญมากในโครงสร้างของการเข้ารหัส มีโอกาสเป็นไปได้สูงที่แฮคเกอร์จะเข้ามายังระบบของกริด และหากมีผู้ใดที่สามารถนำคีย์ส่วนตัวของ ผู้ให้บริการออกใบรับรองออกไปจากระบบได้ จะทำให้ผู้นั้นสามารถแสดงตนเป็นผู้ให้บริการออกใบรับรองได้ทันที การรักษาคีย์ส่วนตัวถือเป็นเรื่องที่สำคัญมาก ดังนั้นจึงต้องดูแลระบบเครือข่ายทางกายภาพ อันได้แก่ การเข้าใช้เครือข่ายระยะไกล และต้องมีการดูแลรวมถึงเก็บข้อมูลเกี่ยวกับการเข้าใช้เครื่องเซิร์ฟเวอร์ด้วย

4.1.3.3. ประเภทของ ใบรับรอง

ใบรับรองที่ใช้ในระบบกริด แบ่งเป็น 2 ประเภท คือ

1. ใบรับรองของผู้ใช้ระบบเป็นหลักฐานเพื่อแยกแยะความแตกต่างของผู้ใช้ระบบกริด
2. ใบรับรองของเครื่องเซิร์ฟเวอร์ เป็นหลักฐานของทางฝั่งเครื่องเซิร์ฟเวอร์

4.1.3.4. การถอดรายชื่อในใบรับรอง

ระหว่างการดำเนินการการพิสูจน์ตัวตน เครื่องคอมพิวเตอร์ภายในระบบกริดจะใช้การพิสูจน์ตัวตนซึ่งกันและกัน(mutual authentication) เพื่อแลกเปลี่ยนใบรับรองดิจิตอลและจะไม่อ้างถึง directory ที่ทำการเก็บใบรับรองอีกต่อไป และกระบวนการนี้เองจึงต้องให้ผู้ใช้ระบบทำการถอนสิทธิ์ของ ใบรับรองเอง

การถอดรายชื่อในใบรับรอง (CRL) จะช่วยชี้ใบรับรองที่ไม่มีความจำเป็นต้องใช้ถึงแม้ว่า ใบรับรองดิจิตอลนั้นยังไม่หมดอายุก็ตาม

4.2.โครงสร้างการรักษาความปลอดภัยในระบบกริด (GSI)

4.2.1. ระบบที่ใช้คีย์สาธารณะ(public key system)

GSI นั้นใช้การเข้ารหัสและถอดรหัสโดยการใช้ คีย์สาธารณะและคีย์ส่วนตัวควบคู่กัน ในการรักษาความปลอดภัยของ GSI นั้นต้องการให้เกิดการพิสูจน์ตัวตนน้อยที่สุด ต้องการการรักษาความถูกต้องบ้าง ซึ่งอาจจะไม่ต้องการเลยก็ได้ในบางกรณี แต่มีความต้องการการรักษาความลับมากที่สุด

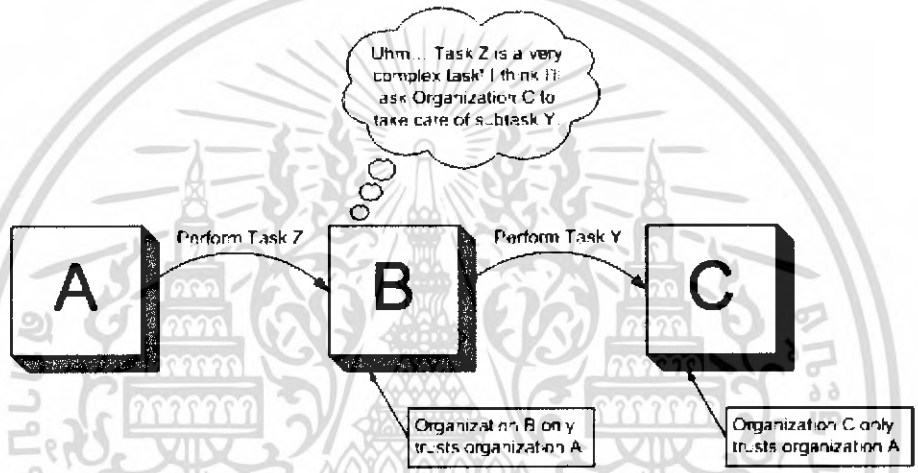
4.2.2. การพิสูจน์ตนซึ่งกันและกัน โดยการใช้ใบรับรอง(Mutual authentication and certificate)

GSI ใช้มาตรฐานของ X.509 และใช้ ใบรับรองในการทำการพิสูจน์ตัวตน และการพิสูจน์ตัวตนซึ่งกันและกันนั้นอาจกล่าวได้ว่า เมื่อเครื่องคอมพิวเตอร์A ต้องการติดต่อกับเครื่องคอมพิวเตอร์ B เครื่องคอมพิวเตอร์A ต้องทรัส(trust) ในเครื่องคอมพิวเตอร์B และเครื่องคอมพิวเตอร์B ต้องทรัสในเครื่องคอมพิวเตอร์A เช่นกัน ซึ่งจะใช้กลไกของการใช้ใบรับรองและผู้ให้บริการออกใบรับรอง

4.2.3. ดีลิเกชั่นและซิงเกิ้ลไซน์ออน (delegation and single-sign-on)

ดีลิเกชั่น หมายถึงการมอบอำนาจหน้าที่ และ ซิงเกิ้ลไซน์ออน จะทำให้ไม่ต้องทำการไซน์เข้าใช้ทรัพยากรหลายครั้ง ซึ่ง 2 คุณลักษณะนี้เป็นคุณลักษณะที่มีความสำคัญมากอย่างหนึ่งของ GSI หรืออาจจะเรียกหน้าที่เหล่านี้ว่า ใบรับรองพรอกซี่ (proxy certificate)

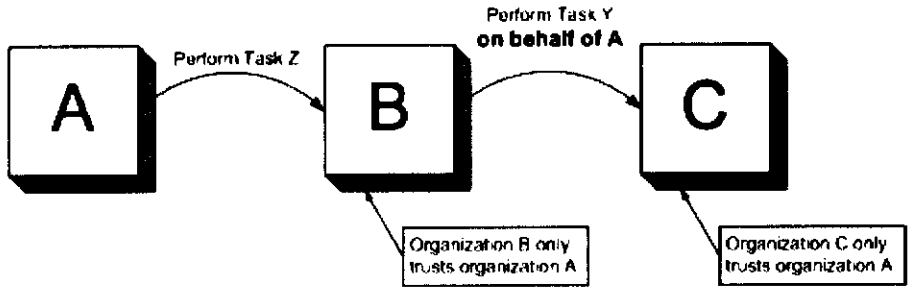
ปัญหาสำคัญที่ทำให้เกิดใบรับรองพรอกซี่



รูปที่ 4-8 สาเหตุของการเกิดพรอกซี่

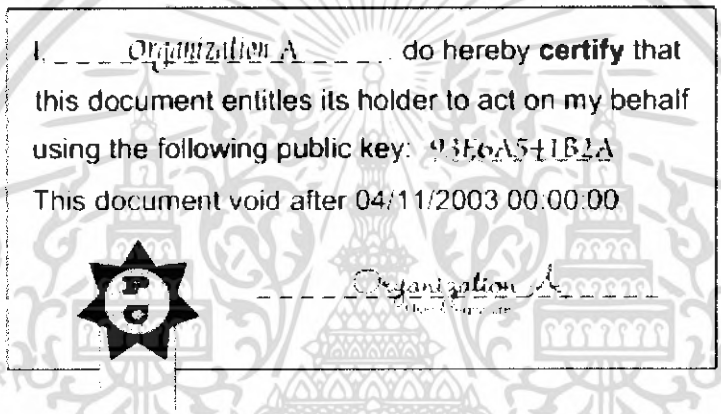
เหตุการณ์จากรูปภาพที่ 4-8 คือ องค์กร A ต้องการให้องค์กร B ปฏิบัติงาน ซึ่งองค์กร B ทรัสองค์กร A และยอมรับที่จะทำงานนั้น แต่งาน Z นั้นมีความซับซ้อนมาก และมีงาน Y ที่เป็นส่วนหนึ่งของงาน Z จะต้องนำไปทำงานที่องค์กรที่สาม ซึ่งได้แก่องค์กร C ในกรณีนี้ องค์กร B จะร้องขอให้องค์กร C ทำงาน Y แต่องค์กร C ไม่ได้ ทรัสองค์กร B องค์กร C ทรัสเพียงองค์กร A เท่านั้น นำไปสู่เหตุการณ์ดังต่อไปนี้

1. ไม่ตอบสนองการร้องขอขององค์กร B เนื่องจากเหตุผลที่ว่า องค์กร C ไม่ทรัสองค์กรB
2. ตอบสนองการร้องขอขององค์กร B เนื่องจากองค์กร C ทรัส องค์กร A และองค์กร C ทราบว่างาน Y ที่ต้องทำนั้นเป็นส่วนหนึ่งของงานจากเครื่อง A ซึ่งแสดงได้จากรูปภาพที่ 4-9 พบว่าเหตุการณ์ดังกล่าวไม่มีความปลอดภัยมากเพียงพอ ถ้าเกิดกรณีที่มีบุคคลอื่นๆปลอมแปลงเป็นองค์กร A การแก้ปัญหานี้ที่มีประสิทธิภาพมาก



รูปที่ 4-9 การทำงานของพรอกซี่

ที่สุดคือทำให้องค์กร C ทรัสว่า องค์กร B คือองค์กร A วิธีที่จะทำให้องค์กร C ทรัสในองค์กร B ได้โดยมีต้องมีการแลกเปลี่ยน คีย์ส่วนตัว และ คีย์สาธารณะ ขององค์กร A ให้กับ B ก็คือ การใช้ใบรับรองพรอกซี่ ซึ่งควรจะมีข้อมูลดังรูปต่อไปนี้



รูปที่ 4-10 ข้อมูลในใบรับรองพรอกซี่

ใบรับรองพรอกซี่ คือเครื่องมือที่ผู้ต้องการมอบอำนาจให้ผู้อื่นสร้างขึ้นและมอบให้กับผู้ที่ต้องการ ทั้งนี้เพื่อแสดงว่าให้อำนาจผู้อื่นในการกระทำการใดๆแทนตนเป็นบางส่วน ใบรับรองพรอกซี่ไม่ได้รับการไซน์จากผู้ให้บริการออกใบรับรอง แต่เป็นการไซน์โดยเอนยูสเซอร์(ผู้ใช้งานปลายทาง)เอง

คีย์แพร์ที่ใช้ในใบรับรองพรอกซี่ นั้นเป็นคีย์แพร์ที่ได้รับการสร้างขึ้นใหม่ระหว่างองค์กร A และ องค์กร B และองค์กร A จะอนุญาตให้องค์กร B เก็บคีย์ส่วนตัวไว้ด้วย

4.2.3.1 กระบวนการสร้าง ใบรับรองพรอกซี่

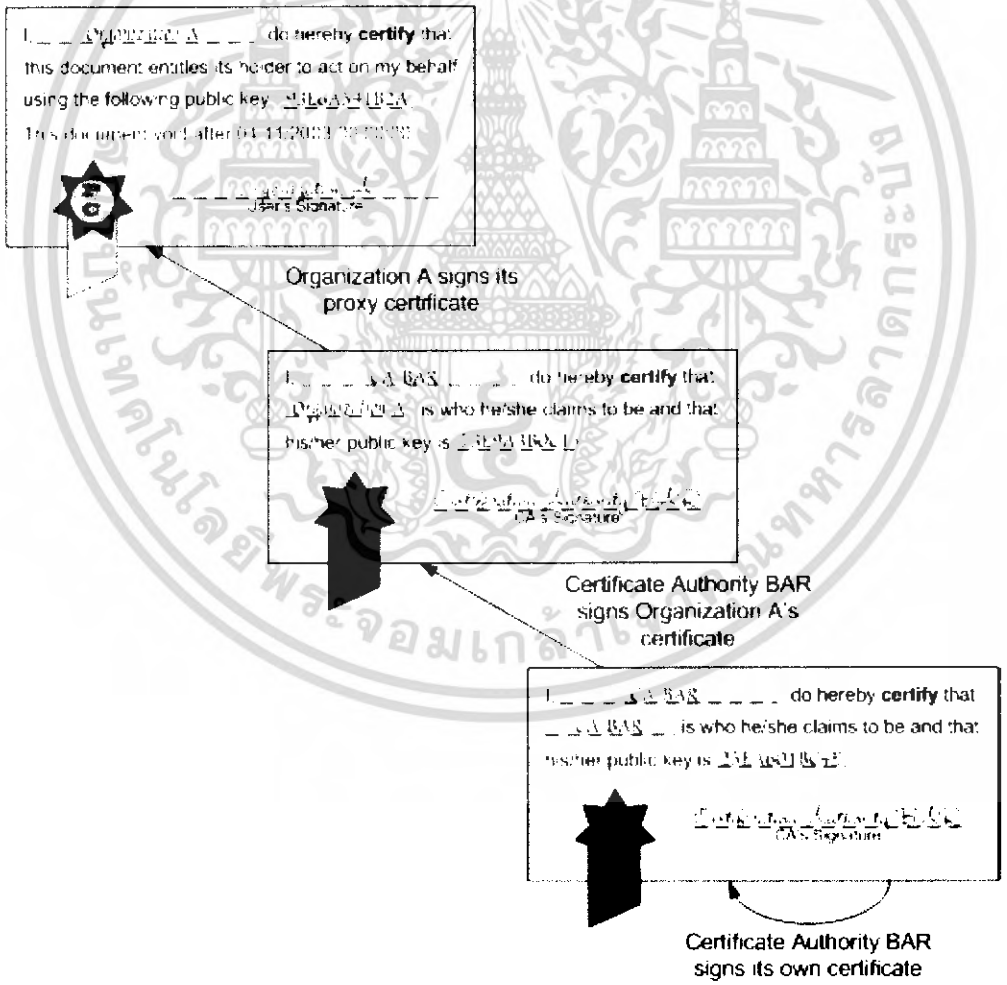
จากเหตุการณ์ข้างต้น พบว่า องค์กรB ต้องการให้องค์กรC ปฏิบัติงาน แต่องค์กรC ไม่ทรัสองค์กรB จึงต้องใช้วิธีการสร้างพรอกซี่เข้ามาช่วยดังนั้น ใบรับรองพรอกซี่ที่ต้องการคือใบรับรองพรอกซี่ที่ได้รับการ ไซน์โดยองค์กรAดำเนินการดังต่อไปนี้

t. องค์กรB สร้างคีย์แพร์ประกอบไปด้วยคีย์ส่วนตัวและคีย์สาธารณะเพื่อใช้ในใบรับรองพรอกซี่

2. องค์กรB ใช้คีย์แพร์เพื่อสร้าง certificate request(คำร้องขอใบรับรอง) และส่งไปยังองค์กรA ซึ่งใน certificate request(คำร้องขอใบรับรอง)จะต้องมีคีย์สาธารณะที่จะใช้ใน ใบรับรองพรอกซี่ แต่จะไม่ส่ง คีย์ส่วนตัวที่สร้างได้จากข้อ 1. ไปด้วย
3. เมื่อองค์กรA ตอบรับเพื่อให้อำนาจส่วนหนึ่งกับองค์กรB แล้ว องค์กรA จะทำการ ไซน์ certificate request(คำร้องขอใบรับรอง) ด้วยคีย์ส่วนตัวขององค์กรA
4. องค์กร Aส่งใบรับรองที่ได้รับการไซน์แล้วกลับไปยังองค์กร B
5. เมื่อองค์กรB ได้รับใบรับรองนั้นแล้ว องค์กรB จะสามารถแสดงตนได้เสมือนกับว่าได้รับอำนาจจากองค์กรA แล้วนั่นเอง และ ใบรับรองพรอกซี่ ที่สร้างขึ้นนี้จะมีอายุกำหนดระยะเวลาที่จะใช้ได้ในช่วงเวลาหนึ่งเท่านั้น

4.2.3.2 กระบวนการใช้งานใบรับรองพรอกซี่

เมื่อองค์กรC ได้รับ ใบรับรองพรอกซี่จากองค์กรB เรียบร้อยแล้วจะต้องมีการตรวจสอบว่าใบรับรองพรอกซี่ ที่ได้รับนั้นมีความถูกต้องหรือไม่ ดังนี้



รูปที่ 4-11 กลไกการรับรอง ใบรับรองพรอกซี่

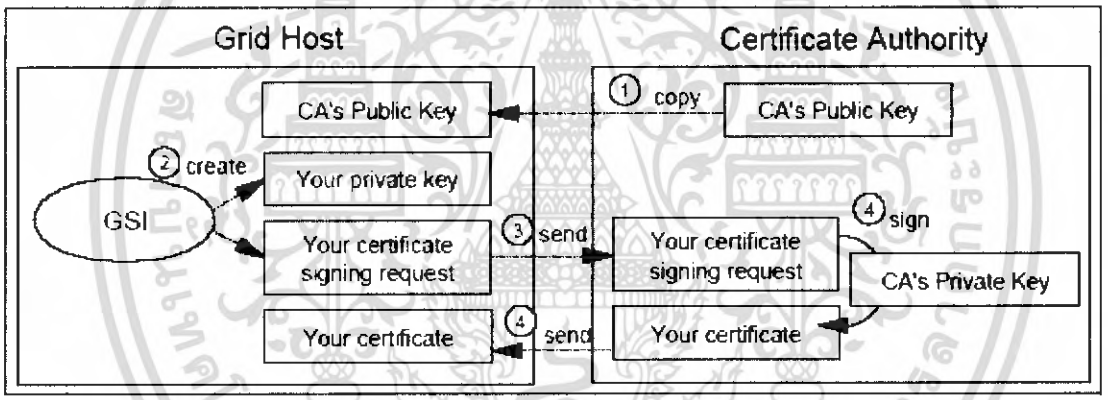
ซึ่งเสมือนการตรวจสอบ ใบรับรองทั่วไปแต่แตกต่างกันเพราะใบรับรองพรอกซี่ไม่ได้เป็นใบรับรองที่ได้รับการไซน์จากผู้ให้บริการออกใบรับรอง ดังนั้นจะต้องเทียบกับคีย์สาธารณะขององค์กร A เท่านั้น

4.3. กระบวนการทำงานในการรักษาความปลอดภัยในระบบกริด (GSI)

4.3.1. การเข้าถึงระบบกริด

มีกระบวนการในการทำงานดังต่อไปนี้

- คัดลอกสำเนาคีย์สาธารณะของผู้ให้บริการออกใบรับรองมาไว้ที่เครื่องคอมพิวเตอร์ที่ต้องการสร้าง GSI
- สร้างคีย์ส่วนตัว และสร้าง certificate request(คำร้องขอใบรับรอง)
- ส่ง certificate request(คำร้องขอใบรับรอง) ไปยังเครื่องเซิร์ฟเวอร์ที่ทำหน้าที่เป็นผู้ให้บริการออกใบรับรอง ด้วย e-mail หรือวิธีการอื่นที่มีความปลอดภัยมากกว่า
- ผู้ให้บริการออกใบรับรอง ยอมรับการร้องของจากเครื่องคอมพิวเตอร์เครื่องนี้ด้วยการสร้างใบรับรองให้และส่งกลับมายังเครื่องที่ทำการร้องขอไป



รูปที่ 4-12 กระบวนการสร้าง ใบรับรองในระบบกริด

เมื่อกระบวนการดังกล่าวข้างต้นเสร็จสิ้นลง ผลที่ได้คือเครื่องคอมพิวเตอร์เครื่องนี้จะได้รับใบรับรองดิจิตอลและจะมีไฟล์ สำคัญ 3 ไฟล์ด้วยกันคือ

- คีย์สาธารณะของผู้ให้บริการออกใบรับรอง
- คีย์ส่วนตัวของเครื่องที่ทำการร้องขอการเข้าใช้กริด
- ใบรับรองดิจิตอลของเครื่องที่ทำการร้องขอการเข้าใช้กริด

เพื่อให้การติดต่อสื่อสารเป็นไปอย่างปลอดภัย จะเพิ่มการรักษาความปลอดภัยของคีย์ส่วนตัวของแต่ละเครื่องอีก 1 ระดับชั้น ด้วยการใช้กลไกของ passphase ซึ่งจะต้องใช้ทุกครั้งที่มีการเรียกใช้คีย์ส่วนตัวร่วมกับใบรับรองดิจิตอล

4.3.2. การพิสูจน์ตัวตนการพิสูจน์สิทธิ์

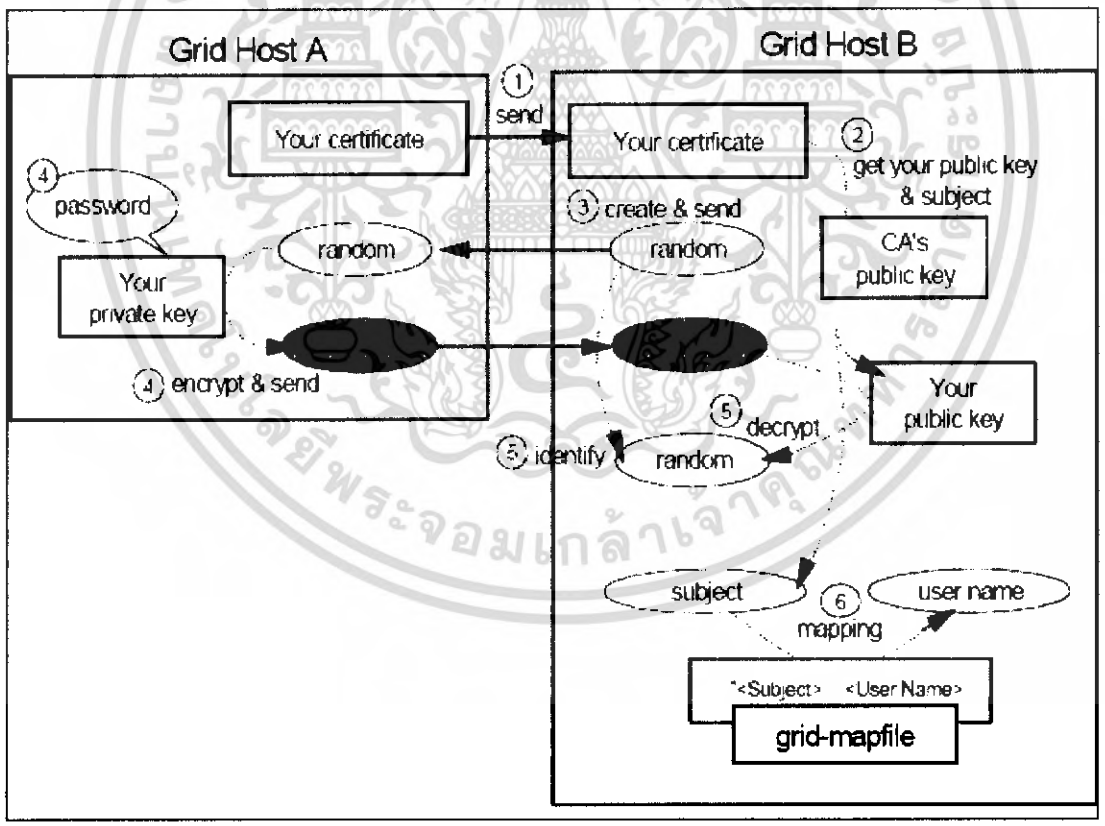
เพื่อให้แน่ใจได้ว่าผู้ที่จะเข้าใช้ทรัพยากรในกริดเป็นผู้ที่ได้รับสิทธิ์นั้นจริงๆ จึงมีกลไก 2 อย่าง ใน GSI ที่รับผิดชอบหน้าที่นี้ คือ การพิสูจน์ตัวตนและการพิสูจน์สิทธิ์

ขั้นตอนในการการพิสูจน์ตัวตนและการพิสูจน์สิทธิ์มีดังนี้

สมมติว่ามีเครื่องคอมพิวเตอร์ 2 เครื่อง คือ A และ B

จากรูปภาพที่ 4-13 แสดงการทำงานของ ใบรับรองพรอกซี่ ในระบบกริดดังต่อไปนี้

1. เครื่อง A ส่งใบรับรองไปยังเครื่อง B เพื่อทำการพิสูจน์ตัวตน
2. เครื่อง B ได้รับคีย์สาธารณะของเครื่อง A และหัวเรื่องของเครื่อง A จากใบรับรองที่ใช้คีย์สาธารณะของผู้ให้บริการออกใบรับรอง
3. เครื่อง B สุ่มเลือกตัวเลขจำนวนใดๆมา 1 จำนวน และส่งให้เครื่อง A
4. เมื่อเครื่อง A ได้รับตัวเลขแล้ว จะทำการเข้ารหัสด้วยคีย์ส่วนตัวของเครื่อง A และส่งข้อมูลที่ได้ทำการเข้ารหัสเรียบร้อยแล้วไปยังเครื่อง B
5. เมื่อเครื่อง B ได้รับข้อมูลแล้ว ทำการถอดรหัสด้วย คีย์สาธารณะ ที่ได้รับมาในข้อที่ 2 และตรวจสอบว่าเป็นตัวเลขที่เคยเลือกไว้ในข้อที่ 3 หรือไม่ หากถูกต้อง เครื่อง A จะได้รับการพิสูจน์ตัวตนสำเร็จ



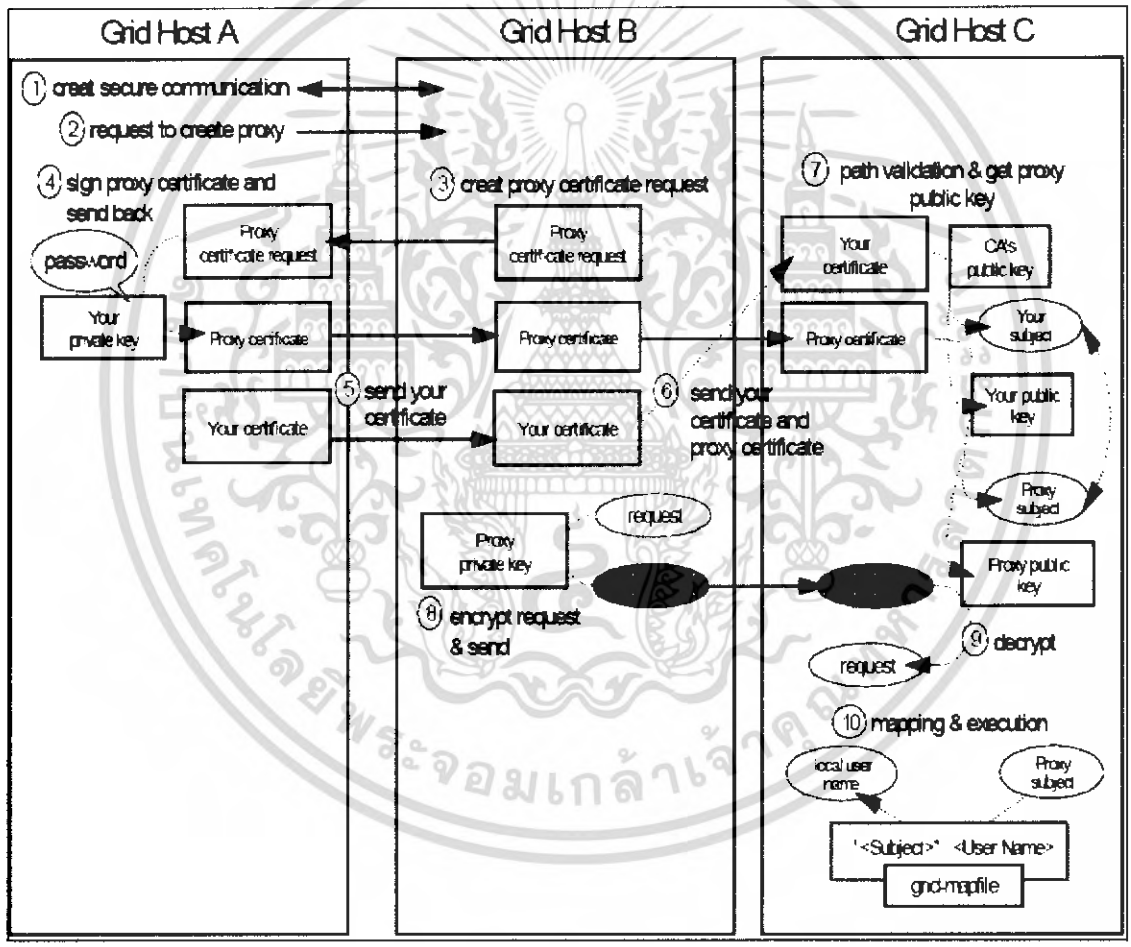
รูปที่ 4-13 กระบวนการทำงานของใบรับรองในระบบกริด

เมื่อการพิสูจน์ตัวตนเรียบร้อยแล้ว ได้รับความเชื่อถือจากเครื่อง B หัวเรื่องของเครื่อง A ในใบรับรองจะถูกจับคู่กับชื่อของผู้ใช้ (local) และหัวเรื่องนั้นจะอยู่ในรูปแบบของ Distinguished Name (DN) เช่น "O=Grid/O=Globus/OU=itso.grid.com/CN=your name"

และจะเป็นหัวเรื่องที่ใช้โดยโปรโตคอล LDAP เพื่อแยกแยะข้อมูลเกี่ยวกับการให้บริการในไดเรกเทอรี่และเสมือนว่า DN ชื่อที่ได้รับการพิสูจน์ตัวตน เรียบร้อยแล้วสามารถเป็นเครื่อง local แม้ว่าจะเป็นเครื่องที่อยู่ในระยะไกลก็ตาม

ในระบบกริดเครื่องคอมพิวเตอร์ใดๆ อาจจะเป็นได้ทั้งเครื่องเซิร์ฟเวอร์ และเครื่องไคล์เอนท์ในเวลาเดียวกัน จะต้องเกิดการพิสูจน์ตัวตนหลายครั้ง ซึ่งเป็นที่ยุ่งยาก จึงเปลี่ยนเป็นวิธีการของการพิสูจน์ตัวตนซึ่งกันและกันแทน เพื่อไม่ต้องทำกระบวนการการพิสูจน์ตัวตนหลายๆครั้ง

4.3.3. พรอกซี่ในGSI



รูปที่ 4-14 กระบวนการทำงานของใบรับรองพรอกซี่ ในระบบกริด

4.3.3.1. กระบวนการในการการพิสูจน์สิทธิ์ด้วยยูสเซอร์พรอกซี่(user proxy)

จากรูป 4-14 มีขั้นตอนการสร้างพรอกซี่ดังนี้

1. สร้างการติดต่อระหว่างเครื่อง A และเครื่อง B ด้วยกระบวนการติดต่อที่เชื่อถือได้

2. เครื่อง A ทำการร้องขอให้เครื่อง B สร้างพรอกซี่เพื่อมอบหน้าที่และสิทธิ์ให้เครื่อง B ทำงานแทนเครื่อง A
3. เครื่อง B สร้างคำร้องขอใบรับรองพรอกซี่(proxy certificate request) ไปยังเครื่อง A
4. เครื่อง A ยอมรับคำร้องขอจากเครื่อง B และสร้างใบรับรองพรอกซี่โดยการไซน์ด้วยคีย์ส่วนตัวของเครื่อง A และส่ง ใบรับรองพรอกซี่ไปยังเครื่อง B

4.3.3.2 การทำงานของพรอกซี่

1. เครื่อง B ส่ง ใบรับรองพรอกซี่ ไปยังเครื่อง C
2. เครื่อง C ได้รับ คีย์สาธารณะ ของ พรอกซี่ (ในที่นี้หมายถึงเครื่อง B) ด้วยการเช็คการมีอยู่ของผู้ใช้ระบบ ด้วยกระบวนการดังนี้
   - เครื่อง C ได้รับหัวเรื่องและคีย์สาธารณะ ของเครื่อง A จาก ใบรับรองโดยการใช้คีย์สาธารณะ ของ ผู้ให้บริการออกใบรับรอง
   - เครื่อง C ได้รับหัวเรื่องของพรอกซี่(หัวเรื่องของเครื่อง B) และ คีย์สาธารณะ ของพรอกซี่(คีย์สาธารณะ ของเครื่อง B) จาก ใบรับรองพรอกซี่ที่ใช้คีย์สาธารณะ ของเครื่อง A
   - หัวเรื่องคือ DN ลักษณะดังนี้
     `"O=Grid/O=Globus/OU=itso.grid.com/CN=your name"`
     และสำหรับหัวเรื่องใน ใบรับรองพรอกซี่จะมีลักษณะ ดังนี้
     `"O=Grid/O=Globus/OU=itso.grid.com/CN=yourname/CN=proxy"` และเพื่อที่จะตรวจสอบ ใบรับรองพรอกซี่ เครื่อง C จะทำการตรวจหาคำว่า CN=proxy จากหัวเรื่องใน ใบรับรองพรอกซี่ ถ้าพบว่ามีเครื่อง B ที่ทำหน้าที่เป็นพรอกซี่จะได้รับการพิสูจน์ตัวตน จากเครื่อง C และ เครื่อง B สามารถทำหน้าที่เสมือนเป็นเครื่อง A
3. พรอกซี่(เครื่อง B)ทำการเข้ารหัสข้อมูลที่มีการร้องขอโดยใช้คีย์ส่วนตัวของพรอกซี่ (เครื่อง B) และส่งกลับไปยังเครื่อง C
4. เครื่อง C ถอดรหัสข้อมูลที่เครื่อง B ส่งมาให้ด้วยคีย์สาธารณะของพรอกซี่ (เครื่อง B)
5. เครื่อง C รันคำร้องขอที่อยู่ภายใต้สิทธิ์ของผู้ใช้ และผู้ใช้จะต้องใช้ mapping file เพื่อจับคู่ระหว่างหัวเรื่องของผู้ใช้ระบบกริด และ ผู้ใช้ที่เป็น local (ชื่อผู้ใช้จริงๆ)

4.3.4. การพิสูจน์ตนซึ่งกันและกัน (Mutual Authentication)

กระบวนการพิสูจน์ตนซึ่งกันและกันนั้นจะเกิดขึ้นเมื่อทรัพยากรภายในกริด 2 แหล่งต้องการจะใช้ทรัพยากรร่วมกันแทนที่จะต้องใช้คีย์ในการพิสูจน์ตัวตน แต่ใช้ใบรับรองดิจิตอลแทน ตัวอย่างเช่น มีทรัพยากรในระบบกริดต้องการสร้างการติดต่อที่มีความปลอดภัยกับทรัพยากรแหล่งอื่น ก่อนที่ผู้รับการร้องขอการติดต่อนั้นจะสามารถเข้าถึงทรัพยากรได้ จะต้องทำการพิสูจน์สิทธิ์ก่อนเสมอ ด้วยวิธีการที่เรียกว่า "SSL handshake"

SSL Handshake

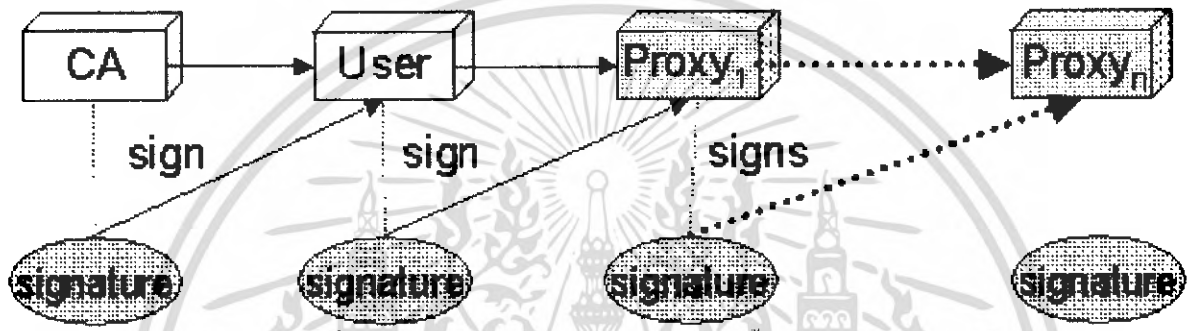
1. เครื่องไคล์เอนท์ในระบบกริดติดต่อไปยังเครื่องเซิร์ฟเวอร์ระยะไกลในระบบกริด เพื่อจะเริ่มต้นสร้างการติดต่อ โดยการใช้ digital X.509 ID Certificate
2. เครื่องไคล์เอนท์ในระบบกริดจะส่งเวอร์ชั่น ของ SSL ที่เครื่องไคล์เอนท์ใช้ , ข้อกำหนดของการเข้ารหัส , ข้อมูลที่สร้างขึ้นด้วยการสุ่ม , และข้อมูลสำคัญอื่นๆที่เครื่องเซิร์ฟเวอร์ต้องการไปให้เครื่องเซิร์ฟเวอร์
3. เมื่อเครื่องเซิร์ฟเวอร์ได้รับข้อมูลในข้อ 2 เรียบร้อยแล้วจะส่ง ใบรับรองดิจิตอลไปยังเครื่องไคล์เอนท์ และยังส่งเวอร์ชั่นของ SSL ที่เครื่องเซิร์ฟเวอร์ใช้ , ข้อกำหนดของการเข้ารหัส , ข้อมูลที่สร้างขึ้นด้วยการสุ่ม , และข้อมูลสำคัญอื่นๆ
4. เครื่องไคล์เอนท์จะต้องพิจารณาข้อมูลที่เครื่องเซิร์ฟเวอร์ส่งมาให้ดังนี้
    a. ตรวจสอบวันหมดอายุของ ใบรับรองของเครื่องเซิร์ฟเวอร์
    b. ใบรับรองของเซิร์ฟเวอร์ฉบับที่ได้ต้องได้รับการไซน์จากผู้ให้บริการออกใบรับรองสร้างเพื่อสำหรับเครื่องไคล์เอนท์ใช้
    c. คีย์สาธารณะของผู้ให้บริการออกใบรับรองสร้างเพื่อเครื่องไคล์เอนท์
    d. Domain name จะถูกระบุโดยเครื่องเซิร์ฟเวอร์และตรงกับ domain name ที่เครื่องเซิร์ฟเวอร์ใช้จริง
5. ถ้าเครื่องเซิร์ฟเวอร์ได้รับการพิสูจน์ตัวตนสำเร็จเรียบร้อยแล้ว เครื่องไคล์เอนท์ในระบบกริดจะสร้าง เซสชั่นคีย์ (session key) เพื่อใช้ในการเข้ารหัสข้อมูลทั้งหมดตลอดการติดต่อ
6. เครื่องไคล์เอนท์ในระบบกริดทำการเข้ารหัสเซสชั่นคีย์ด้วยคีย์สาธารณะ ของเครื่องเซิร์ฟเวอร์และส่งไปยังเครื่องเซิร์ฟเวอร์
7. เครื่องเซิร์ฟเวอร์ถอดรหัสเซสชั่นคีย์ด้วยคีย์ส่วนตัว
8. เครื่องไคล์เอนท์ส่งข้อมูลไปยังเครื่องเซิร์ฟเวอร์เพื่อบอกว่าในอนาคตเครื่องไคล์เอนท์จะทำการเข้ารหัสด้วยเซสชั่นคีย์และเครื่องเซิร์ฟเวอร์ส่งข้อมูลไปยังเครื่องไคล์เอนท์เพื่อบอกว่าในอนาคตจะเข้ารหัสด้วยเซสชั่นคีย์เช่นเดียวกัน
9. SSL-Secure session สร้างการติดต่อขึ้น และ SSL ใช้คีย์เพียงคีย์เดียวในการเข้ารหัสและถอดรหัส
10. เมื่อมีทรัพยากรแหล่งที่หนึ่งได้รับการพิสูจน์ตัวตนดังนั้นทรัพยากรแหล่งที่สองจะทำการพิสูจน์ตนด้วยกระบวนการเดียวกัน
11. เมื่อทำการยกเลิกการติดต่อเซสชั่นคีย์จะถูกระงับการใช้ไป

4.3.5. การทำดีลิเกชั่นและซิงเกิ้ลไซน์ออนใน GSI

GSI ได้จัดเตรียมความสามารถด้านการดีลิเกชั่นไว้ซึ่งได้ขยายมาจากโปรโตคอลมาตรฐาน SSL เพื่อลดจำนวนเวลาที่ผู้ใช้จะต้องใส่ passphrase ถ้าการคิดแบบกริดนั้นต้องการหลายรีซอร์สที่เคยใช้ (ที่

ต้องการในการรับรองทั้ง 2 ทาง) หรือต้องการผู้แทน (local หรือ remote) ที่ร้องขอเซอร์วิสที่เป็นประโยชน์กับผู้ใช้ ความจำเป็นที่จะให้ใส่ passphrase ของผู้ใช้อีกครั้งนั้น สามารถหลีกเลี่ยงได้ด้วยการสร้างพร็อกซี่

พร็อกซี่ เกิดขึ้นจากใบรับรองใหม่(ที่มีคีย์สาธารณะข้างในมัน) และมีคีย์ส่วนตัวใหม่ โดยที่ใบรับรองใหม่อันนั้นจะเป็นค่าของเจ้าของ โดยส่วนที่ดัดแปลงให้ต่างออกไปจากเดิมนั้นคือ พร็อกซี่ ซึ่งใบรับรองใหม่จะถูกลงชื่อโดยเจ้าของ มากกว่าจากผู้ให้บริการออกใบรับรองจากรูป 4-15 ใบรับรองจะรวมค่าเวลาซึ่ง พร็อกซี่ไม่ควรจะถูกยอมรับโดยส่วนอื่นๆด้วย กล่าวคือ พร็อกซี่นั้นต้องมีไลฟ์ไทม์ที่จำกัด



รูปที่ 4-15 แสดงวิธีการดีลิเกชั่นและซิงเกิ้ลไซน์ออน

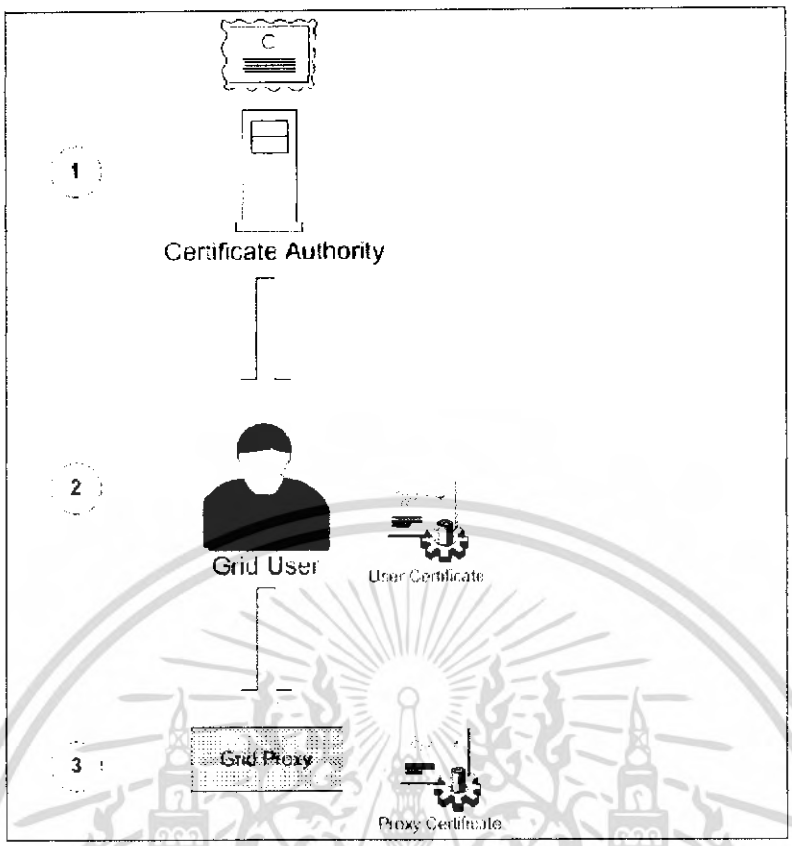
คีย์ส่วนตัวของพร็อกซี่ต้องเก็บเป็นความลับ แต่เพราะพร็อกซี่ไม่ได้ถูกต้องในเวลาที่นานมากนัก ทำให้มันไม่ต้องเก็บให้ปลอดภัยเท่ากับค่าคีย์ส่วนตัวของเจ้าของ ซึ่งเป็นไปได้ที่จะเก็บค่าคีย์ส่วนตัวของ พร็อกซี่บนแหล่งเก็บข้อมูลแบบโลคอลโดยปราศจากการเข้ารหัส เท่าที่การอนุญาตของไฟล์จะต้องการปกป้องเพื่อไม่ให้หาได้โดยง่าย เมื่อพร็อกซี่สร้างและเก็บไว้แล้ว ผู้ใช้จะใช้พร็อกซี่สำหรับรับรองการสื่อสารทั้ง 2 ทางโดยปราศจากการใส่พาสเวิร์ด

เมื่อพร็อกซี่ถูกใช้ กลุ่มที่อยู่ระยะไกลนั้นจะไม่ได้รับเพียงแค่ใบรับรองของพร็อกซี่ แต่จะได้รับใบรับรองของเจ้าของด้วย ระหว่างการรับรองทั้ง 2 ทางนั้น คีย์สาธารณะของเจ้าของจะถูกใช้เพื่อตรวจสอบลายมือชื่อของใบรับรองพร็อกซี่ โดยคีย์สาธารณะของผู้ให้บริการออกใบรับรอง จะถูกใช้เพื่อตรวจสอบใบรับรองของเจ้าของ สิ่งนี้จะสร้างโซ่แห่งความเชื่อมั่นสำหรับพร็อกซี่ผ่านไปยังเจ้าของได้

4.4. ขั้นตอนการจัดการระบบความปลอดภัยของระบบกริด

4.4.1. Local delegation

สำหรับ globus toolkit มีคำสั่ง grid-proxy-init เพื่อทำการสร้างพรอกซี่ ซึ่งก่อนที่จะทำงานใดๆในระบบกริดจะต้องสร้าง พรอกซี่ ก่อนซึ่ง จากคำสั่ง grid-proxy-init นี้ จะรวมกระบวนการทำซิงเกิ้ลไซน์ออนด้วย



รูปที่ 4-16 กระบวนการพิสูจน์ตน

4.4.2. ลำดับของการสร้าง พรอกซี่

1. ทรัพยากรของกริดที่อยู่ระยะไกลนั้นจะต้องได้รับการเชื่อถือจาก ผู้ให้บริการออกใบรับรอง ซึ่งจะทำการใส่ ผู้ให้บริการออกใบรับรอง จะเก็บ certificate(ใบรับรอง)ไว้ที่ path /etc/grid-security/certificates
2. ทรัพยากรของกริดที่อยู่ระยะไกลนั้นสามารถการพิสูจน์ตัวตนของผู้เรียกใช้ด้วย ใบรับรองของผู้ใช้คนนั้น เพราะว่ามีดิจิตอลซิกเนเจอร์ที่รับประกันโดย ผู้ให้บริการออกใบรับรอง
3. ทรัพยากรของกริดที่อยู่ระยะไกลนั้นสามารถการพิสูจน์ตัวตนกับพรอกซี่ได้ด้วยใบรับรองของผู้ใช้คนนั้น

บทที่ 5

Parallel Programming

การเขียนโปรแกรมแบบขนาน เป็นเทคนิคการเขียนโปรแกรมที่ใช้สำหรับการ execute operation พร้อมๆกัน ทั้งภายในเครื่องๆเดียว หรือข้ามระบบจำนวนหนึ่ง

ในการเขียนโปรแกรมแบบขนาน งานเดี่ยวหลายๆงานจะถูกแตกออกเป็นงานย่อยจำนวนหนึ่งที่สามารถถูกประมวลผลแยกจากกันอย่างเป็นอิสระ และถูกนำกลับมารวมกันเพื่อสร้างผลลัพธ์ที่ถูกต้อง

การเขียนโปรแกรมแบบขนานทำให้เราสามารถทำงานที่มีขนาดใหญ่และซับซ้อนให้เร็วขึ้นได้ จากการที่เรามีงาน (task) ขนาดใหญ่เราสามารถแบ่งย่อยเป็นงานที่มีขนาดเล็กกว่าและทำงานไปพร้อมๆกันได้ สิ่งที่จำเป็นในการทำงานแบบขนานคือ

- การแบ่งงาน (task) ออกเป็นงานย่อยๆ
- การมอบหมายงานไปยังเครื่องที่ทำงานเพื่อทำงานพร้อมกัน
- การติดต่อสื่อสารเพื่อทำงานร่วมกันระหว่างเครื่องที่ทำงาน

การเขียนโปรแกรมแบบขนานจะเอาไปใช้แก้ปัญหา (solving) ต่างๆ เช่น การสร้างตึก, การทำงานในองค์กรขนาดใหญ่, การสร้างเครื่องจักรผลิตรถยนต์

5.1 ลักษณะของ parallel computer

ลักษณะของ parallel computer แบ่งออกเป็น 4 หมวดดังต่อไปนี้

5.1.1. Single Instruction, Single Data (SISD)

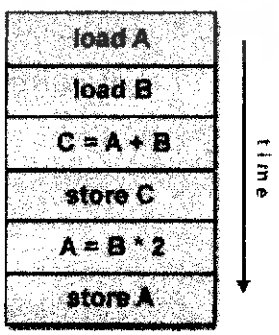
ไม่ใช่คอมพิวเตอร์แบบขนาน

ทำงานทีละคำสั่งเท่านั้นในแต่ละ clock cycle

มีการใช้งานข้อมูลอันเดียวเท่านั้นต่อหนึ่ง clock cycle

Deterministic execution

ประสิทธิภาพวัดเป็น MIPS (million of instructions per second) หรือ clock frequency เป็น MHz



รูปที่ 5-1 Single Instruction, Single Data (SISD)

5.1.2. Single Instruction, Multiple Data (SIMD)

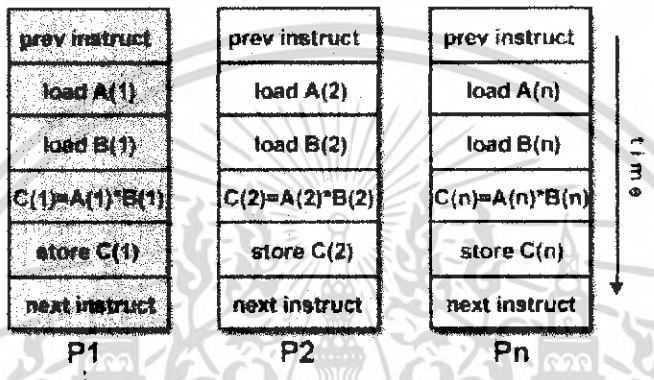
เป็นสถาปัตยกรรมแบบ von Neumann

คำสั่งแต่ละคำสั่งจะทำงานกับข้อมูลมากกว่าหนึ่งตัว

มี intermediate host ทำการ run โปรแกรมและ broadcast คำสั่งไปยังโปรเซสเซอร์ตัวอื่นๆ

Synchronous (lockstep)

ประสิทธิภาพวัดเป็น MFLOPS (million of floating point operations per second)



รูปที่ 5-2 Single Instruction ,Multiple Data (SIMD)

5.1.3. Multiple Instruction, Single Data (SIMD)

ในความเป็นจริงนั้นไม่ค่อยมีตัวอย่างของการทำงานแบบนี้

ตัวอย่างที่เป็นไปได้คือ filter ที่มีหลายความถี่ทำงานบน single signal stream และ cryptography algorithms หลายแบบซึ่งพยายามที่จะ crack รหัส

5.1.4. Multiple Instruction, Multiple Data (SIMD)

สามารถทำงานแบบขนานได้โดยการต่อโปรเซสเซอร์หลายๆตัวเข้าด้วยกัน

โปรเซสเซอร์แต่ละตัว execute คำสั่งของตัวมันเองไม่ขึ้นต่อโปรเซสเซอร์ตัวอื่นๆ

โปรเซสเซอร์แต่ละตัวทำงานกับข้อมูลที่แตกต่างกัน

สามารถ executions แบบ synchronous หรือ asynchronous, deterministic หรือ non-deterministic

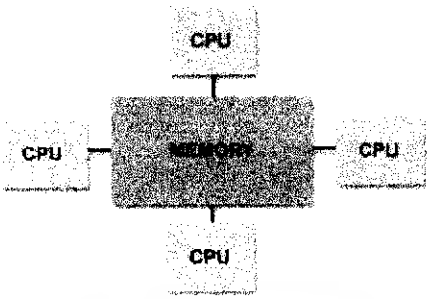
มีปัญหา overhead ในการทำ load balancing ในการทำงานแบบ synchronization

ยากต่อการเขียนโปรแกรม

เป็นรูปแบบของซูเปอร์คอมพิวเตอร์ในปัจจุบันและคอมพิวเตอร์แบบขนานที่เชื่อมต่อกันด้วยเครือข่าย (กริด)

5.2. Parallel Computer Memory Architectures

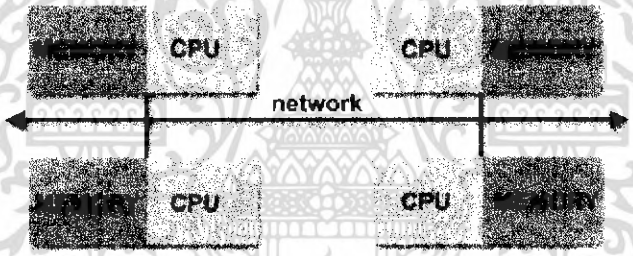
5.2.1. Shared Memory



รูปที่ 5-3 Shared Memory

โปรเซสเซอร์แต่ละตัวจะใช้หน่วยความจำเดียวกัน และ อ้างถึงหน่วยความจำโดยใช้ global address space ข้อมูลที่ถูกแก้ไขโดยโปรเซสเซอร์ตัวหนึ่งจะมีผลต่อโปรเซสเซอร์ตัวอื่นๆ แบบนี้จะง่ายต่อการพัฒนาโปรแกรม และมีความเร็วในการอ้างถึงข้อมูลสูง

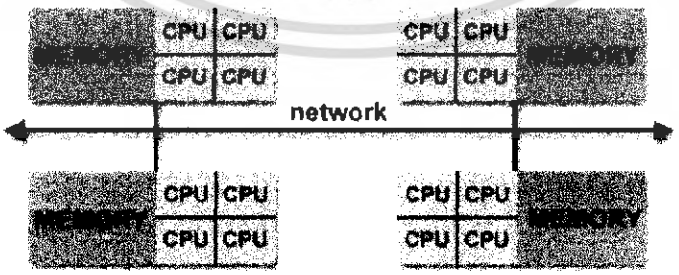
5.2.2. Distributed Memory



รูปที่ 5-4 Distributed Memory

โปรเซสเซอร์แต่ละตัวจะทำงานแบบอิสระต่อกันและมีหน่วยความจำเป็นของตนเอง การแชร์ข้อมูลจะกระทำโดยผ่านการติดต่อทางเครือข่ายและใช้การทำ message passing ซึ่งรูปแบบนี้จะไม่มี overhead ในการดูแล cache coherency

5.2.3. Hybrid Distributed-Shared Memory



รูปที่ 5-5 Hybrid Distributed-Shared Memory

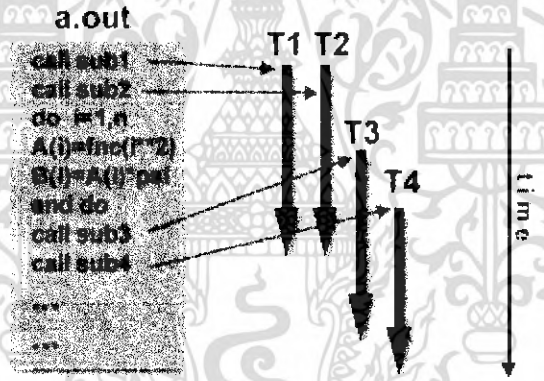
5.3 Parallel Programming Paradigms

5.3.1 Shared Memory Model

ในโมเดลของการเขียนโปรแกรมแบบแชร์นั้นงานจะถูกแชร์ address ที่ใช้ในการอ่านและ เขียน โดยจะมีกระบวนการต่างๆในการควบคุมการเข้าถึง shared memory เช่น locks และ semaphores ข้อ ได้เปรียบของโมเดลนี้จากมุมมองของโปรแกรมเมอร์คือไม่ต้องทำการติดต่อข้อมูลระหว่างงาน (task) ทำให้ง่ายในการพัฒนาโปรแกรม แต่จะมีปัญหาในการทำความเข้าใจและจัดการเรื่องของ data locality ในการ implement โมเดลนี้จะใช้ compiler เปลี่ยนตัวแปรของ user program ไปเป็นค่า actual memory addresses ที่เป็นแบบ global

5.3.2 Threads Model

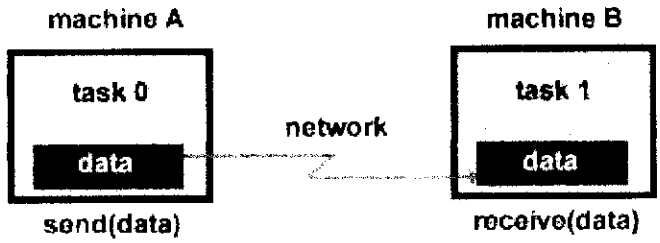
ในโมเดลแบบ thread โปรเซสหนึ่งโปรเซสจะสามารถมี path ที่ทำการ execution พร้อมกันได้ หลาย path ยกตัวอย่างเช่นโปรแกรม a.out ถูกกำหนดให้ run โดยระบบปฏิบัติการณ์ a.out จะถูกโหลด และจะถือครอง system และ user resources ที่จำเป็นในการ run ไว้ จากนั้น a.out จะสร้าง thread ที่ สามารถรันโดยระบบปฏิบัติการณ์พร้อมๆกัน ในแต่ละ thread จะมีข้อมูลของตัวเอง thread จะ ติดต่อสื่อสารกันผ่านทาง global memory ในการ implement โมเดลนี้จะใช้ POSIX Threads และ OpenMP



รูปที่ 5-6 Threads Model

5.3.3 Message Passing Model

โมเดลแบบนี้แต่ละงานจะมีหน่วยความจำของตัวเองในระหว่างการคำนวณ งานหลายๆงาน สามารถอยู่บนเครื่องเดียวกันหรือต่างเครื่องกันก็ได้ งานจะแลกเปลี่ยนข้อมูลโดยการรับส่ง message ซึ่ง อาศัยการทำงานร่วมกันของทั้งสองโปรเซสยกตัวอย่างเช่น โปรเซสหนึ่งกำลังส่งโปรเซสอีกโปรเซสก็ ต้องทำการรับข้อมูล



รูปที่ 5-7 Message Passing Model

ในการ implement โมเดลนี้เราจะใช้ MPI (Message Passing Interface) ซึ่งเป็นมาตรฐานสำหรับการติดต่อแบบส่งข้อความ (Message Passing) นิยมใช้มากในการคำนวณแบบขนาน (Parallel Computing) และผู้ขาย (Vender) ทั่วไป MPI เป็นมาตรฐานที่ไม่ขึ้นกับแพลตฟอร์ม (Platform-Independent) และไม่ขึ้นกับภาษาที่ใช้เขียน (Language-Independent) สำหรับ library ของการส่งข้อความ (Message-Passing Library) นั้นได้รับการพัฒนาโดย MPI Forum ซึ่งเป็นการร่วมมือกันระหว่างผู้ขายคอมพิวเตอร์แบบขนาน, ผู้เขียน library และผู้เชี่ยวชาญด้านแอพพลิเคชัน (Application Specialist) และ library ของ MPI นี้จะใช้ภาษา Fortran และ C เป็นหลัก

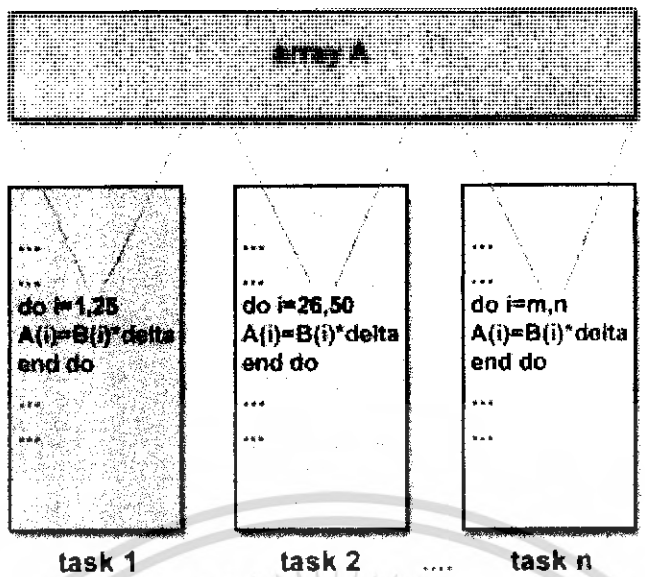
สาเหตุที่ทำให้ MPI เป็นที่นิยมนั้นเป็นเพราะว่ารายละเอียดของมันไม่ขึ้นกับผู้ขาย, ระบบปฏิบัติการ, ฮาร์ดแวร์ และภาษาที่ใช้ ซึ่งหมายความว่า โปรแกรมใดก็ตามที่เขียนโดยมีพื้นฐานมาจาก MPI (Based on MPI) นั้นสามารถที่จะคอมไพล์(Compile)และ run ได้โดยที่ไม่ต้องเปลี่ยน Source Code บน PC จากผู้ขายหรือระบบปฏิบัติการใดๆ

นอกจาก MPI แล้วยังมี library ที่สำคัญคือ PVM ซึ่งเป็นโปรแกรมแบบขนานเช่นเดียวกัน โดยได้มีการพัฒนามาก่อน library ของ MPI แต่ปัจจุบันนี้นิยมใช้ library ของ MPI มากกว่า เนื่องจากมีฟังก์ชันการทำงานที่มากกว่า PVM และใช้ได้ง่ายกว่าด้วย

5.3.4 Data Parallel Model

โมเดลนี้จะมีคุณสมบัติดังนี้คือ งานแบบขนานส่วนใหญ่จะทำงานบนเซตของข้อมูล โดยข้อมูลจะถูกจัดโครงสร้างธรรมดาๆ เช่น array หรือ cube เซตของงานเก็บบนโครงสร้างข้อมูลเดียวกัน อย่างไรก็ตามแต่ละงานจะทำงานบน partition ที่แตกต่างกันบนโครงสร้างข้อมูลเดียวกัน (data structure)

ในสถาปัตยกรรมแบบ shared memory ทุกๆงานจะเข้าถึงข้อมูลโดยผ่าน global memory ส่วนสถาปัตยกรรมแบบ distributed memory ข้อมูลจะถูกแบ่งเป็น chunks ใน local memory ของแต่ละงาน



รูปที่ 5-8 Data Parallel Model

5.4 ขั้นตอนการสร้างโปรแกรมแบบขนาน

การเขียนโปรแกรมแบบขนานมีขั้นตอนวิธีการพัฒนาดังต่อไปนี้

5.4.1 ถ้าเราสร้างโปรแกรมขนาน (parallel program) จากโปรแกรมอนุกรม (serial program)

เราควรจะ Debug โปรแกรมอนุกรมให้เรียบร้อยก่อน จากนั้นเราควรศึกษา algorithm ด้วยว่าสามารถแก้ปัญหาโดยวิธี parallel programming ได้หรือเปล่า ยกตัวอย่างเช่น

ตัวอย่างโจทย์ที่แก้ได้โดย parallel programming:

คำนวณพลังงานศักย์ของ conformation หลายพันอันที่เป็นอิสระจากกันของแต่ละโมเลกุล เมื่อเสร็จแล้วหาค่า conformation ของพลังงานที่น้อยที่สุด

จากโจทย์ข้อมูลแต่ละอันเป็นอิสระต่อกันดังนั้นเราจึงสามารถคำนวณไปพร้อมๆกันได้จึงแก้ปัญหาโดยใช้ parallel programming ได้

ตัวอย่างโจทย์ที่แก้ไม่ได้ด้วย parallel programming:

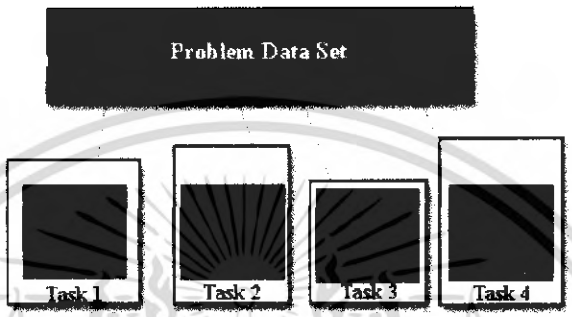
การคำนวณ Fibonacci series (1,1,2,3,5,8,13,21,...) โดยใช้สูตรต่อไปนี้

$F(k+2) = F(k+1) + F(k)$

โจทย์นี้เราจะเห็นได้ว่าการที่เราจะบวกค่าได้ เราต้องอาศัยข้อมูลก่อนหน้านี้ ดังนั้นเราไม่สามารถจะทำการบวกพร้อมๆกันได้ จึงไม่สามารถใช้ parallel programming ในการแก้ปัญหาได้

5.4.2 ทำการระบุว่าส่วนไหนของโปรแกรมสามารถที่จะทำงานพร้อมกันได้

5.4.3 ทำการ Decompose ออกตามฟังก์ชันหรือข้อมูล



รูปที่ 5-9 Decompose Data Set

5.4.3.1 Function Decomposition (Functional Parallelism)

แบ่งปัญหาออกเป็นหลายๆส่วนเพื่อกระจายไปยังหลายๆ processor เพื่อทำการ execution พร้อมกัน ซึ่งแบบนี้จะเหมาะกับปัญหาที่มีโครงสร้างไม่คงที่และมีจำนวนการคำนวณที่แน่นอน

5.4.3.2 Domain Decomposition (Data Parallelism)

แบ่งขอบเขตข้อมูลของปัญหาและกระจายส่วนต่างๆไปยังหลายๆ processor เพื่อทำการ execution พร้อมกัน เหมาะแก่การใช้กับปัญหาซึ่ง

ข้อมูลมีค่าคงที่ (การ factor และแก้ไขปัญหา matrix ที่มีขนาดใหญ่หรือจำนวนการคำนวณที่จำกัด)

โครงสร้างข้อมูลที่เป็นแบบ dynamic ถูกรวมเป็น entity เดียวซึ่ง entity สามารถถูกแบ่งเป็น subset ย่อยๆได้ (ปัญหาแบบ multi-body ที่มีขนาดใหญ่)

ขอบเขตมีค่าจำกัดแต่การคำนวณในหลายๆส่วนของขอบเขตจะเป็นแบบ dynamic (fluid vortices models)

5.4.4 พัฒนาโค้ด (Code development)

5.4.5 compile, ทดสอบโปรแกรม, และ Debug

5.4.6 Optimization คือการวัดประสิทธิภาพ, กำหนดขอบเขตของปัญหา และปรับปรุงให้ดีขึ้น

5.5 สิ่งที่ควรพิจารณาในการสร้างโปรแกรมแบบขนาน

5.5.1 Amdahl's Law

Amdahl's Law ระบุว่าความเร็วของโปรแกรมจะถูกกำหนดโดยส่วนของ code (P) ที่สามารถทำงานแบบขนานได้

$$Speedup = 1/(1 - P)$$

ถ้าไม่มีส่วนใดของ code เลยที่สามารถทำงานแบบขนานแล้ว P = 0 แล้ว speedup = 1 (ไม่มี speedup) ถ้า code ทั้งหมดสามารถทำงานแบบขนานได้แล้ว P = 1 แล้ว speedup มีค่าเป็น infinity (ตามทฤษฎี)

ถ้า 50% ของ code สามารถทำงานแบบขนานได้แล้ว ค่า speedup สูงสุดจะเท่ากับ 2 ซึ่งหมายความว่า code จะ run เร็วขึ้นเป็น 2 เท่า

การหาจำนวนของ processor ที่ทำงานในส่วนที่ทำงานแบบขนานนั้น มีความสัมพันธ์ดังนี้

$$Speedup = 1 / ((P / N) + S)$$

เมื่อ P = parallel fraction (ส่วนที่ทำงานแบบขนาน)

N = จำนวนของโปรเซสเซอร์

S = serial fraction (ส่วนที่ทำงานแบบอนุกรม)

มีข้อจำกัดของ scalability ในการทำงานแบบขนาน ตัวอย่างเช่น เมื่อ

P = .50, .90 และ .99 (50%, 90% และ 99% ของ code ที่สามารถทำงานแบบขนานได้)

	speedup		
N	P = .50	P = .90	P = .99
10	1.82	5.26	9.17
100	1.98	9.17	50.25
1000	1.99	9.91	90.99
10000	1.99	9.91	99.02

ตารางที่ 5-1 ข้อจำกัดของ scalability ในการทำงานแบบขนาน

อย่างไรก็ตามสามารถแสดงประสิทธิภาพที่เพิ่มขึ้นโดยการเพิ่มขนาดของปัญหาตัวอย่างเช่น

2D Grid Calculations	85 วินาที	85%
Serial fraction	15 วินาที	15%

ตารางที่ 5-2 ประสิทธิภาพที่เพิ่มขึ้นโดยการเพิ่มขนาดของปัญหา

เราสามารถเพิ่มขนาดของปัญหาโดยการแบ่งครึ่ง grid points และ time step ซึ่งเป็นสัดส่วนโดยตรงกับ grid spacing ผลลัพธ์นี้เป็น 4 เท่าของจำนวน grid point (2 ส่วนในแต่ละทิศทาง) และ 2 เท่าของจำนวน time step การใช้เวลาเป็นดังนี้

2D Grid Calculations	680 วินาที	97.84%
Serial fraction	15 วินาที	2.16%

ตารางที่ 5-3 เพิ่มขนาดของปัญหาโดยการแบ่งครึ่ง grid points และ time step

ปัญหาซึ่งมีเปอร์เซ็นต์ของ parallel time มากขึ้นเกิดจากขนาดของมันที่มีความยืดหยุ่น (scalable) มากกว่าปัญหาที่มีเปอร์เซ็นต์ของ parallel time จำกัด

5.5.2 Load Balancing

Load balancing จะรับรองว่าการกระจายงานนั้นใช้เวลาในการ execution แบบขนานอย่างมีประสิทธิภาพที่สุด

ถ้างานถูกกระจายอย่างไม่สมดุลจะจัดการโดยการหยุดรอเพื่อทำงานหนึ่งให้เสร็จก่อนในขณะที่งานอื่นๆมีสถานะว่าง

ประสิทธิภาพสามารถเพิ่มขึ้นได้ถ้างานถูกกระจายมากขึ้นอย่างเช่น ถ้ามีงานจำนวนมากซึ่งมีขนาดที่หลากหลาย มันจะเพิ่มประสิทธิภาพได้โดยการดูแล task pool และกระจายงานไปยัง โปรเซสเซอร์เมื่อโปรเซสเซอร์แต่ละตัวทำงานเสร็จ

พิจารณาสภาพแวดล้อมที่ต่างกันเมื่อมีเครื่องจำนวนมากที่มีกำลังในการประมวลผลและ user load ที่ต่างกันทำงานในสภาพแวดล้อมเดียวกันคือมีโปรเซสเซอร์เหมือนกันโดยจะทำการ run งานหนึ่งงานต่อโปรเซสเซอร์หนึ่งตัว

5.5.3 Granularity

เพื่อที่จะให้โปรเซสเซอร์ที่ต่างกันทำงานร่วมกันในการแก้ปัญหาเดียวกัน ซึ่งจะต้องมีการกำหนดรูปแบบในการสื่อสารระหว่างโปรเซสเซอร์

อัตราส่วนระหว่างการประมวลผลและการสื่อสารจะถูกเรียกว่า granularity

granularity ที่มีประสิทธิภาพที่สุดนั้นจะขึ้นอยู่กับอัลกอริทึมและฮาร์ดแวร์ที่ใช้ทำงาน

ส่วนใหญ่ overhead ที่สัมพันธ์กับการสื่อสารและการ synchronization มีค่าสูงเมื่อเทียบกับความ เร็วในการ execute ซึ่งเป็นข้อดีของ coarse granularity

5.5.3.1 Fine-grain parallelism

- งานทั้งหมดจะ execute คำสั่งจำนวนหนึ่งระหว่าง cycle ในการสื่อสาร

- อัตราส่วนระหว่างการประมวลผลกับการสื่อสารมีค่าต่ำ
- ใช้ประโยชน์จาก load balancing
- มี overhead ในการสื่อสารสูงและโอกาสในการเพิ่มประสิทธิภาพต่ำ
- ถ้า granularity ละเอียดมากเกินไปจะต้องการ overhead ในการสื่อสารและการ synchronization ระหว่างงานใช้เวลามากกว่าเวลาในการประมวลผล

5.5.3.2 Coarse-grain parallelism

เป็นการประมวลผลแบบยาวซึ่งประกอบด้วยคำสั่ง (instruction) จำนวนมากระ หว่างจุด synchronization ในการสื่อสาร

อัตราส่วนระหว่างการประมวลผลกับการสื่อสารมีค่าสูง

มีโอกาสในการเพิ่มประสิทธิภาพสูง

ยากต่อการทำ load balancing อย่างมีประสิทธิภาพ

5.5.4 Data Dependency

- data dependency เกิดขึ้นเมื่อมีข้อมูลจำนวนมากใช้ storage location เดียวกัน
- สิ่งสำคัญของdependency คือการป้องกันการทำ parallel execution อย่างสม่ำเสมอ
- ตัวอย่างเช่น

```
DO 500 J = MYSTART,MYEND
A( J) = A( J - 1 ) * 2.0
500 CONTINUE
```

จะเห็นว่า code นี้มี data dependency โดยต้องมีค่าที่ถูกประมวลผลสำหรับ A(J-1) ก่อนที่เราจะสามารถคำนวณ A(J) และถ้างานที่ 2 มี A(J) แล้วงานที่ 1 มี A(J-1) โดยค่าของ A(J) ขึ้นอยู่กับว่าเป็น Distributed memory หรือ Shared memory

Distributed memory

งานที่ 2 ได้รับค่าของ A(J-1) จากงานที่ 1

Shared memory

งานที่ 2 อ่าน A(J-1) ก่อนหรือหลังจากงานที่ 1 จะ update มันก็ได้

วิธีในการจัดการ data dependency

Distributed memory ทำการเชื่อมต่อข้อมูลที่ต้องการที่จุด synchronization

Shared memory ทำการ synchronize ในการอ่าน / เขียนระหว่างงานหลายๆงาน

5.5.5 Deadlock

- Deadlock อธิบายเงื่อนไขที่โปรเซส 2 โปรเซส หรือมากกว่านั้นกำลังรอ event หรือการสื่อสารจากโปรเซสอื่น

- ตัวอย่างที่ง่ายที่สุดถูกแสดงโดย 2 โปรเซสซึ่งถูกเขียนโปรแกรมเพื่ออ่าน / รับค่าจากอีกโปรเซสหนึ่งก่อนที่จะทำการเขียน / ส่ง

Task 1	Task 2
X = 4	Y = 8
SOURCE = TASK2 RECEIVE (SOURCE,Y)	SOURCE = TASK1 RECEIVE (SOURCE,X)
DEST = TASK2 SEND (DEST,X)	DEST = TASK1 SEND (DEST,Y)
Z = X + Y	Z = X + Y

ตารางที่ 5-4 ตัวอย่างการทำงานที่เกิด deadlock

วิธีการแก้ปัญหาวิธีหนึ่งคือการเปลี่ยนลำดับของ SEND และ RECEIVE ในส่วนหนึ่งของงาน

งานที่ 1	งานที่ 2
X = 4	Y = 8
SOURCE = TASK2 RECEIVE (SOURCE,Y)	DEST = TASK1 SEND (DEST,Y)
DEST = TASK2 SEND (DEST,X)	SOURCE = TASK1 RECEIVE (SOURCE,X)
Z = X + Y	Z = X + Y

ตารางที่ 5-5 การแก้ปัญหา deadlock โดยการเปลี่ยนลำดับของ SEND และ RECEIVE

วิธีแก้ปัญหาวิธีอีกวิธีหนึ่งคือการใช้ NON-BLOCKING message passing

5.5.6 รูปแบบในการสื่อสารและแบนด์วิธ

ในบางปัญหานั้นการเพิ่มจำนวนของโปรเซสเซอร์จะลด execution time ในการประมวลผล แต่เพิ่ม execution time ในการสื่อสาร

เวลาที่ใช้ในการสื่อสารขึ้นอยู่กับ bandwidth ที่ใช้ในการสื่อสาร

ตัวอย่างเช่น เวลา (t) ที่ต้องการเพื่อส่ง W words ระหว่างโปรเซสเซอร์ 2 ตัวคือ

$$t = L + W/B$$

โดยที่ L = latency

B = hardware bitstream rate เป็น word ต่อวินาที

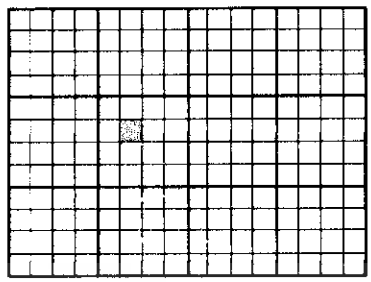
Latency นั้นสามารถคิดว่าเป็นเวลาที่ต้องการเพื่อส่ง zero byte message

- รูปแบบการสื่อสารจะส่งผลกระทบต่ออัตราส่วนของการประมวลผลต่อการสื่อสาร ตัวอย่างเช่น การสื่อสารแบบ gather-scatter ระหว่างโปรเซสเซอร์ตัวหนึ่งกับโปรเซสเซอร์อื่น N ตัวจะได้รับผลกระทบจากการเพิ่ม latency มากกว่าโปรเซสเซอร์ N ตัวที่เชื่อมต่อเพียงแค่กับ neighbor ที่ใกล้ที่สุด

5.5.7 I/O Patterns

- I/O operation มักจะถูกมองว่าเป็นตัวขัดขวางการทำงานแบบขนาน
- ระบบ I/O แบบขนานนั้นยังไม่ถูก define และยังไม่มีในปัจจุบัน
- ในสภาพแวดล้อมที่ processor ทั้งหมดเห็น filespace เดียวกันนั้นการเขียนจะทำให้เกิดการ overwrite file
- ความสามารถของ fileserver ส่งผลกระทบต่อการอ่านคือจะทำการร้องขอในการอ่านหลายๆอันในเวลาเดียวกัน
- I/O ซึ่งต้องถูกดำเนินการบนเครือข่าย (non-local) อาจจะทำให้เกิดปัญหา bottleneck
- วิธีการอื่นๆ ได้แก่
    - ลด I/O ทั้งหมดเท่าที่จะเป็นไปได้
    - กำหนดขอบเขตของ I/O ให้เป็นส่วนที่ทำงานแบบ serial โดยเฉพาะ ตัวอย่างเช่น งานที่ 1 สามารถอ่าน input file และสื่อสารข้อมูลที่ต้องการให้กับงานอื่นๆและงานที่ 1 สามารถทำการเขียนหลังจากได้รับข้อมูลที่ต้องการจากงานอื่นทั้งหมดแล้ว
    - สร้าง filename เฉพาะสำหรับ input/output file ของแต่ละงาน
    - สำหรับระบบหน่วยความจำแบบกระจายที่มีการแชร์ filespace จะทำงาน I/O ใน filespace ที่เป็นแบบ local และ non-shared ตัวอย่างเช่น แต่ละโปรเซสเซอร์สามารถใช้ /tmp filspace ซึ่งมีประสิทธิภาพมากกว่าการทำงาน I/O บนเครือข่ายไปยัง home directory ของเครื่องแต่ละเครื่อง

```
do j = 1,n
do i = 1,n
  a(i,j) = fcn(i,j)
end do
end do
```



รูปที่ 5-10 Array Processing

5.6 ตัวอย่างโปรแกรมแบบขนาน (Parallel Example)

5.6.1 Array Processing

ตัวอย่างนี้จะแสดงการคำนวณบนอาร์เรย์ 2 มิติ ซึ่งข้อมูลในแต่ละส่วนไม่ขึ้นต่อกัน ถ้าเป็นการคำนวณจากโปรแกรมแบบอนุกรม (serial program) จะมี source code ดังนี้

5.6.1.1 Array Processing Parallel Solution 1

```
find out if I am MASTER or WORKER

if I am MASTER

 initialize the array
 send each WORKER info on part of array it owns
 send each WORKER its portion of initial array

 receive from each WORKER results

else if I am WORKER
 receive from MASTER info on part of array I own
```

```
receive from MASTER my portion of initial array

# calculate my portion of array
do j = my first column,my last column
do i = 1,n
  a(i,j) = fcn(i,j)
end do
end do

send MASTER results

endif
```

จาก Source Code ข้างต้นจะมีตัวประมวลผลอยู่สองประเภทคือ Master และ Worker ซึ่งตัว master จะเป็นคนแจกจ่ายขอบเขตของงานไปยัง worker และรอรับผลลัพธ์ในการทำงานกลับมา ส่วน worker จะนำเอางานที่ได้จาก master มาคำนวณแล้วส่งผลลัพธ์กลับไป

5.6.1.2 Array Processing Parallel Solution 2 :Pool of Tasks

```
find out if I am MASTER or WORKER

if I am MASTER

 do until no more jobs
  send to WORKER next job
  receive results from WORKER
 end do

 tell WORKER no more jobs

else if I am WORKER
```

```
do until no more jobs
 receive from MASTER next job

 calculate array element: a(i,j) = fcn(i,j)

 send results to MASTER
end do

endif
```

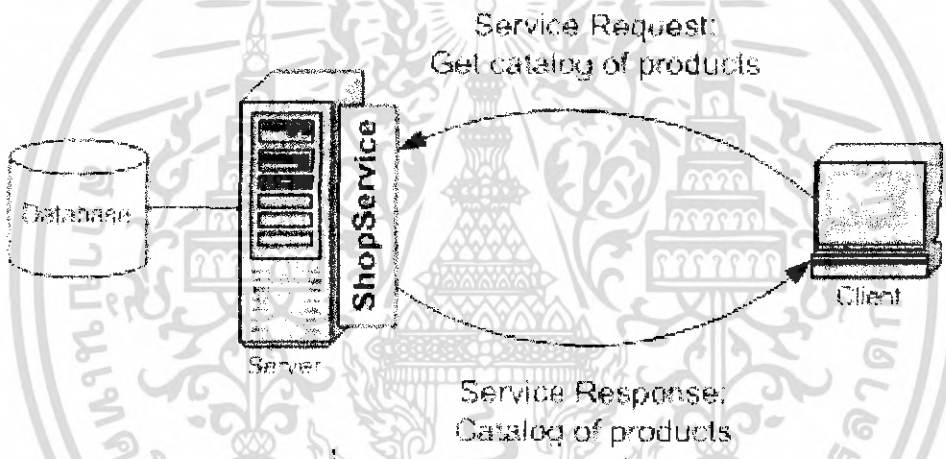
จากแบบที่ 1 เราจะเจอปัญหาในเรื่องของ load balancing เนื่องจากเรามีการแจกจ่ายงานแบบ static ดังนั้นถ้าเครื่อง worker ตัวไหนสามารถทำงานได้เร็วกว่า ก็จะมีเวลาว่างเหลือ ซึ่งการแก้ปัญหานี้เราจะใช้วิธี Pool of Tasks จะเป็นการแจกจ่ายงานทีละงานไปยังเครื่องที่ว่างอยู่เท่านั้น

บทที่ 6
กริดเซอร์วิส

6.1 เว็บเซอร์วิส

เป็นเทคโนโลยีการคำนวณแบบกระจายที่สร้างแอพพลิเคชั่นในไคล์เอนท์/เซิร์ฟเวอร์ (client/server) เช่น สมมติว่า เราต้องพัฒนาแอพพลิเคชั่นสำหรับโซ่คลังสินค้าที่มีอยู่ทั่วประเทศแต่ละชื่อสินค้าอยู่ในฐาน ข้อมูลที่สำนักงานกลางเท่านั้น ดังนั้นซอฟท์แวร์ที่คลังสินค้าเหล่านั้นต้องสามารถเข้าถึงรายชื่อสินค้าที่สำนัก งานกลางได้ ทำได้โดยผ่านเว็บเซอร์วิสที่ชื่อ ShopService (ไม่ควรให้ใช้บนเว็บไซต์ทั่วไป เพราะทุกคนสามารถเห็นสิ่งเหล่านั้นได้ ซึ่งต่างจากการใช้เว็บเซอร์วิสเพราะเป็นการเรียกใช้ผ่านซอฟท์แวร์)

ไคล์เอนท์ที่ต้องการใช้เว็บเซอร์วิสจะส่งการร้องขอไปยังเซิร์ฟเวอร์ เพื่อขอรายการต่างๆจากนั้นเซิร์ฟเวอร์ก็จะส่งรายการนั้นมาให้เป็นการตอบรับของเซอร์วิส ดังภาพด้านล่าง



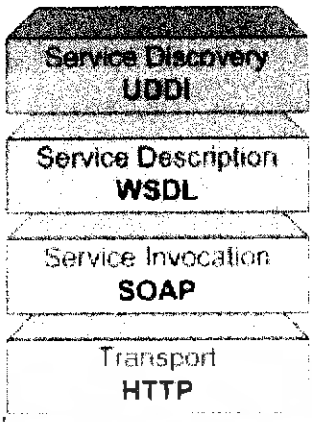
รูปที่ 6-1 การตอบรับของเซอร์วิส

ข้อดีของเว็บเซอร์วิส ประการแรกคือ ไม่ขึ้นกับแพลตฟอร์มและไม่ขึ้นกับภาษาที่ใช้เขียน เนื่องจากใช้ภาษามาตรฐาน XML ซึ่งหมายความว่าเว็บเซอร์วิสที่เขียนด้วยภาษา JAVA บน Linux Server สามารถทำงานบนเครื่อง Windows ที่มีโปรแกรมภาษา C++ ประการที่สองคือ เว็บเซอร์วิส เกือบทั้งหมดใช้ HTTP ในการรับ-ส่งข้อมูล (เช่น การร้องขอ และ ตอบกลับของเซอร์วิส) เป็นผลดีต่อการทำแอพพลิเคชั่นที่ใช้งานบนอินเตอร์เน็ต เนื่องจาก HTTP ไม่มีปัญหากับพรอกซี่และไฟร์วอล

6.1.1 โครงสร้างของเว็บเซอร์วิส

จากรูปที่ 6-2 แสดงให้เห็นโครงสร้างของเว็บเซอร์วิสที่มีองค์ประกอบดังนี้

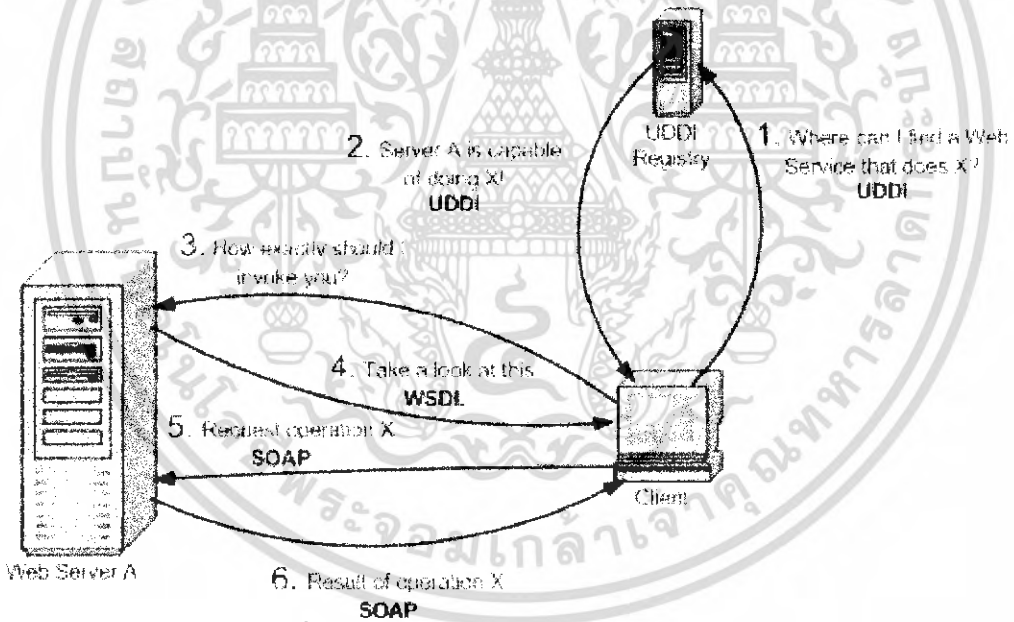
- Service Discovery : ช่วยให้สามารถทราบเว็บเซอร์วิสที่เหมาะกับความต้องการโดยใช้ UDDI
- Service Description : เมื่อรู้ว่าจะติดต่อ เว็บเซอร์วิส ได้ที่ไหนแล้ว เราก็สามารถขอให้ เว็บเซอร์วิส นั้นอธิบายอธิบายตัวเองในแง่ของแอพพลิเคชั่นที่ใช้, คำสั่งที่มี, วิธีเรียกใช้ โดย WSDL



รูปที่ 6-2 โครงสร้างของเว็บเซอร์วิส

- Service Invocation : การเรียกใช้ เว็บเซอร์วิส รวมไปถึงการส่งข้อความระหว่างไคล์เอนท์และเซิร์ฟเวอร์มักจะใช้ SOAP ในการกำหนดรูปแบบที่ใช้
- Transport : ใช้ HTTP ในการรับ-ส่ง ระหว่างไคล์เอนท์กับเซิร์ฟเวอร์

6.1.2 การเรียกใช้ เว็บเซอร์วิส



รูปที่ 6-3 แสดงลำดับการเรียกใช้เว็บเซอร์วิส

จากรูปที่ 6-3 แสดงให้เห็นขั้นตอนในการเรียกใช้เว็บเซอร์วิส

1. ไคล์เอนท์ไม่จำเป็นต้องมีความรู้เกี่ยวกับเว็บเซอร์วิสที่ตัวเองกำลังจะเรียกใช้มาก่อนโดยจะเริ่มต้นจากค้นหาเว็บเซอร์วิสที่เหมาะสมกับความต้องการของเราโดยติดต่อไปยัง UDDI registry
2. UDDI จะตอบกลับมาว่าเซิร์ฟเวอร์ไหนมีเซอร์วิสที่เราต้องการ

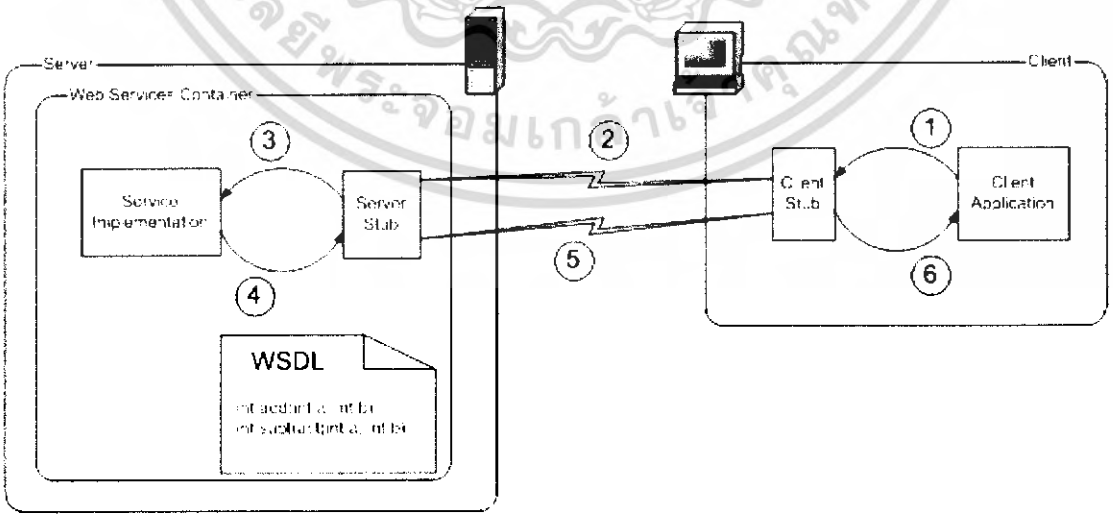
3. เมื่อรู้เว็บเซอร์วิสแล้วแต่ก็ยังไม่รู้ว่าจะเรียกใช้อย่างไรต้องถามไปยังเว็บเซอร์วิส นั้นให้อธิบายวิธีเรียกใช้
4. เว็บเซอร์วิสจะส่ง WSDL มาให้
5. เมื่อรู้เว็บเซอร์วิสและวิธีเรียกใช้ก็จะเกิดการเรียกใช้ขึ้นเองโดยภาษา SOAP ส่งการร้องขอไปยังเว็บเซอร์วิส
6. เว็บเซอร์วิส ตอบกลับมาด้วย SOAP response พร้อมสิ่งที่เราต้องการ หรืออาจจะเป็นข้อความแสดงความผิดพลาด ถ้า SOAP ที่ส่งไปนั้นผิด

6.1.3 แอพพลิเคชั่นของเว็บเซอร์วิส

เมื่อไคล์เอนท์ต้องการเรียกใช้เว็บเซอร์วิสเราสามารถสร้างทาสก์ (task) แทรกไว้ใน ซอฟท์แวร์เรียกว่า clicnt stub สำหรับเรียกใช้เว็บเซอร์วิสแทนที่จะเขียน SOAP หรือ WSDL ทีละบรรทัด ซึ่งหลายๆ เครื่องมือในปัจจุบันมีการสร้าง client stub ให้อย่างอัตโนมัติ โดยยึดเอา WSDL เป็นพื้นฐาน ดังนั้นขั้นตอนที่ถูกต้องในการเรียกใช้เว็บเซอร์วิส เป็นดังนี้

1. เรารู้ที่ตั้งของเว็บเซอร์วิสที่เหมาะสมกับความต้องการผ่าน UDDI
2. เราได้รับ WSDL ของ เว็บเซอร์วิส นั้น
3. เราสร้าง stub ขึ้นมาแทรกไว้ในแอพพลิเคชั่น
4. แอพพลิเคชั่นจะใช้ stub นั้นในแต่ละครั้งที่ต้องการเรียกใช้เว็บเซอร์วิส

เราสามารถสร้างฟังก์ชั่นทั้งหมดของ เว็บเซอร์วิส ขึ้นมา แล้วสร้าง server stub จาก WSDL เพื่อใช้แปลการร้องขอและส่งต่อให้เซอร์วิสที่สร้างขึ้นมา และเมื่อเซอร์วิสนั้นได้ผลลัพธ์ก็จะส่งไปให้ server stub ซึ่งจะทำการสร้าง SOAP response ที่เหมาะสม ในการจัดการกับเซอร์วิสและ server stub จะใช้ ผ่านซอฟท์แวร์ที่เรียกว่า เว็บเซอร์วิสคอนเทนเนอร์ (Web Service container) เพื่อให้แน่ใจว่า HTTP request ที่จะเข้ามายัง เว็บเซอร์วิส ได้เข้ามาที่ server stub ได้อย่างถูกต้อง



รูปที่ 6-4 ลำดับการเรียกใช้เว็บเซอร์วิส

สมมติว่า เรารู้ที่ตั้ง เว็บเซอร์วิส แล้ว และสร้าง client stubs จาก WSDL และ โปรแกรมทางฝั่งเซิร์ฟเวอร์ก็สร้าง server stubs แล้ว สามารถอธิบายขั้นตอนการเรียกใช้ เว็บเซอร์วิส ได้ดังนี้

1. เมื่อไคล์เอนท์ต้องการเรียกใช้เว็บเซอร์วิสก็จะทำการเรียก client stub ซึ่งจะทำการแปลงคำขอให้เป็น SOAP request เรียกขั้นตอนนี้ว่า marshaling หรือ serializing process
2. SOAP request ถูกส่งผ่านเครือข่ายโดยใช้ผ่านโปรโตคอล HTTP จากนั้นเว็บเซอร์วิสคอนเทนเนอร์ ก็จะได้รับ SOAP request และส่งให้ server stub ซึ่งจะทำการเปลี่ยน SOAP request ให้อยู่ในรูปที่เซอร์วิสสามารถเข้าใจได้ เรียกขั้นตอนนี้ว่า unmarshaling หรือ deserializing
3. เซอร์วิสได้รับการร้องขอ จาก server stub และทำงานตามการร้องขอนั้น
4. ผลลัพธ์ที่ได้ถูกส่งไปยัง server stub และถูกแปลงเป็น SOAP response
5. SOAP response ถูกส่งผ่านเครือข่ายผ่านโปรโตคอล HTTP เมื่อ client stub ได้รับก็จะทำการแปลง SOAP response นั้นให้อยู่ในรูปที่แอพพลิเคชั่นเข้าใจได้
6. แอพพลิเคชั่นได้รับผลลัพธ์ที่ต้องการจากเว็บเซอร์วิสไปใช้

6.2 กริดเซอร์วิส

กริดเซอร์วิส เป็นส่วนขยายของเว็บเซอร์วิสที่ขยายขึ้นตามมาตรฐานที่กำหนดไว้ใน Open Grid Service Infrastructure เพื่อนำมาใช้งานในการสร้างระบบกริด ในหัวข้อต่อไปนี้กล่าวถึงมาตรฐานที่ใช้กำหนดกริดเซอร์วิส ให้ทราบก่อน

6.2.1 OGSA (Open Grid Service Architecture)

เป็นสิ่งที่กำหนดโดย The Global Grid Forum เพื่อใช้กำหนดมาตรฐานแอพพลิเคชั่นของกริด และเพื่อให้สามารถใช้งานได้จริง เนื่องจากกริดเซอร์วิสเป็นเทคโนโลยีการคำนวณแบบกระจายภายใต้ OGSA จึงกล่าวได้ว่า OGSA เป็นรากฐานของกริดเซอร์วิส

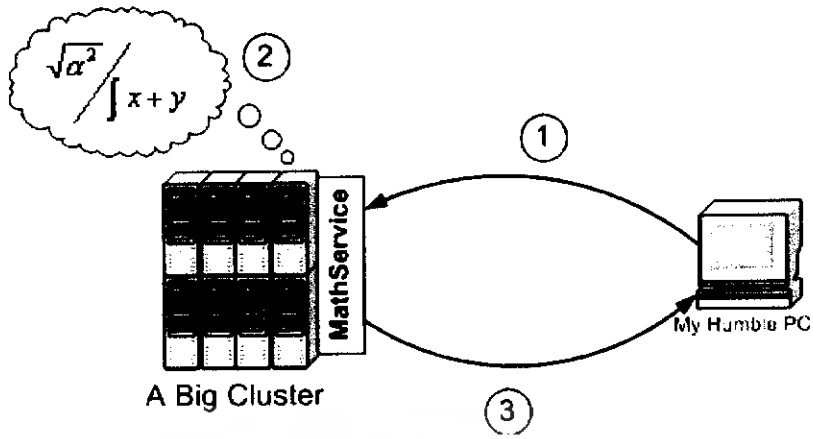
6.2.2 OGSI (Open Grid Service Infrastructure)

เนื่องจาก OGSA เป็นเพียงโครงร่างของ กริดเซอร์วิส แต่ไม่ได้อธิบายรายละเอียดถึงขั้นการใช้งานจริง จึงได้พัฒนาให้มี OGSI ที่ใช้ระบุวิธีการทำงานของ กริดเซอร์วิส โดยละเอียด

6.2.3 คุณสมบัติหลักที่ได้รับการพัฒนาใน OGSI

- Stateful and potentially transient services

  เป็นลักษณะสำคัญอย่างหนึ่งที่เกี่ยวข้องกับเว็บเซอร์วิสยกตัวอย่างเพื่อความเข้าใจดังนี้ สมมติถ้าเรามีกลุ่มคลัสเตอร์หรือคอมพิวเตอร์ ขนาดใหญ่ที่สามารถประมวลผลงานยากๆได้ ตั้งอยู่ที่สำนักงานในกรุงเทพ และต้องการให้พนักงานที่เชียงใหม่, ขอนแก่น และชลบุรี สามารถใช้งานคลัสเตอร์นี้ได้อย่างสะดวก โดยที่เราอาจจะสร้าง เว็บเซอร์วิสที่ใช้คำนวณทางคณิตศาสตร์ขึ้นมา เรียกว่า MathService ซึ่งมีคำสั่ง SolveReallyBigSystem(), SolveFermatsLastTeorem()

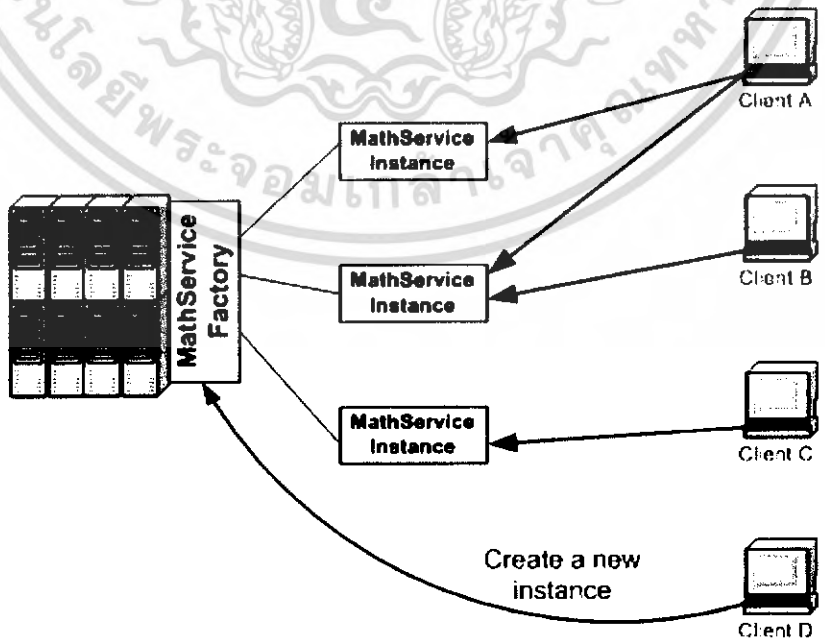


รูปที่ 6-5 ลำดับการทำงานแบบมีสเตทของ OGSI

และอื่นๆให้ใช้งานการทำงานของเซอร์วิสนี้จะเป็นตามขั้นตอนดังนี้

1. เรียกใช้ MathService โดยขอใช้คำสั่งต่างๆของเซอร์วิสนั้น
2. MathService จะสั่งให้กลุ่มคลัสเตอร์ทำงานตามคำสั่งที่ขอไว้
3. MathServicc จะคืนผลลัพธ์ที่ได้กลับมาให้

อย่างไรก็ตาม หากว่ากันตามจริงแล้ว เราอาจต้องการให้คลัสเตอร์ทำงานให้เราหลายคำสั่ง ซึ่งมีความเกี่ยวข้องกันในคราวเดียว ถ้าเป็นเว็บเซอร์วิสซึ่งเป็น stateless แล้วล่ะก็ จะไม่สามารถจดจำได้ว่าการเรียกใช้แต่ละคำสั่งได้ ว่ามีความสัมพันธ์กันอย่างไร แต่ในกริดเซอร์วิสนั้น จะแก้จุดนั้นโดยใช้แฟคทอรี่ (factory) ที่เมื่อไคล์เอนท์ต้องการสร้างหรือลบอินสแตนซ์ (instance) ต้องบอกผ่านแฟคทอรี่อินสแตนซ์ มีอายุการใช้งานชั่วคราวเท่าที่ไคล์เอนท์ต้องการเพื่อจะได้ไม่ต้องขึ้นกับอายุการใช้งานของกริดเซอร์วิสคอนเทนเนอร์แต่จะขึ้นอยู่กับแอพพลิเคชั่นแต่ใช่ว่าทุกกริดเซอร์วิส จะต้องใช้ แฟคทอรี่ หรือ อินสแตนซ์เท่านั้น แต่กริดเซอร์วิสสามารถรอให้บริการ



รูปที่ 6-6 แสดงการเรียกใช้ MathService Instance

ตลอดเวลาเหมือนเว็บเซอร์วิสทั่วไปได้ตามแต่สมควรกับแอพพลิเคชั่นนั้นๆ

- Lifecycle management

  การจัดการกับวัฏจักรชีวิตของเซอร์วิสโดยตรงนั้นเป็นเรื่องยาก (ถ้าใช้ผ่านแฟคทอรี่หรืออินสแตนซ์เนื่องจากอินสแตนซ์จะถูกสร้างและลบทิ้งเมื่อไหร่ก็ได้)จึงต้องมี lifecycle management มาช่วยจัดการในเกี่ยวกับวัฏจักรชีวิตของกริดเซอร์วิส

- Service Data

  Service Data มีความเป็น stateful และใช้งานได้แบบชั่วคราว เป็นสิ่งสำคัญที่ กริดเซอร์วิส มีเพิ่มจาก เว็บเซอร์วิส เพื่อช่วยให้เราสามารถเพิ่มโครงสร้างข้อมูล เข้าไปในทุกๆเซอร์วิส ทำให้สามารถเข้าถึงคำสั่งต่างๆได้โดยตรงผ่านอินเตอร์เฟสของมันเอง โดยข้อมูลที่เพิ่มเข้าไปมักเป็น

  - State information : ให้ข้อมูลสเตทปัจจุบันของเซอร์วิส
  - Service metadata : ข้อมูลเกี่ยวกับตัวเซอร์วิสเอง

- Notifications

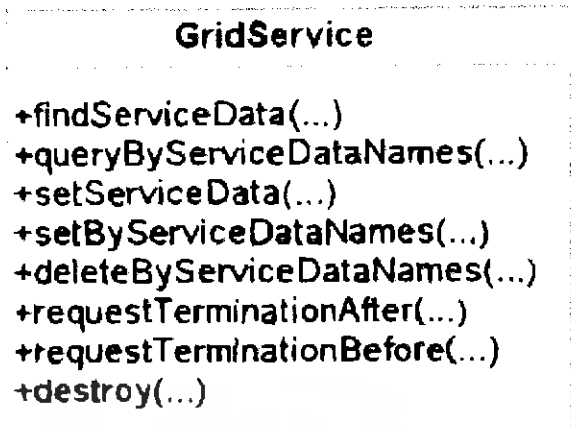
  สามารถใช้ กริดเซอร์วิส เป็นแหล่งประกาศ (notification) และไคล์เอนท์เป็นผู้รับประกาศนั้น เมื่อมีการเปลี่ยนแปลงบางอย่างเกิดขึ้นกับกริดเซอร์วิสจะมีการแจ้งไปยังไคล์เอนท์ทุกตัว (แจ้งเฉพาะการเปลี่ยนแปลงบางอย่างตามแต่ผู้เขียนโปรแกรมของกริดเซอร์วิสต้องการ)

- Service Groups

  เราสามารถนำเซอร์วิส ต่างๆมาจับกลุ่มเข้าไว้ด้วยกัน แล้วเรียกใช้ได้ในครั้งเดียว โดยสามารถเพิ่มหรือลบเซอร์วิสออกจากกลุ่มได้

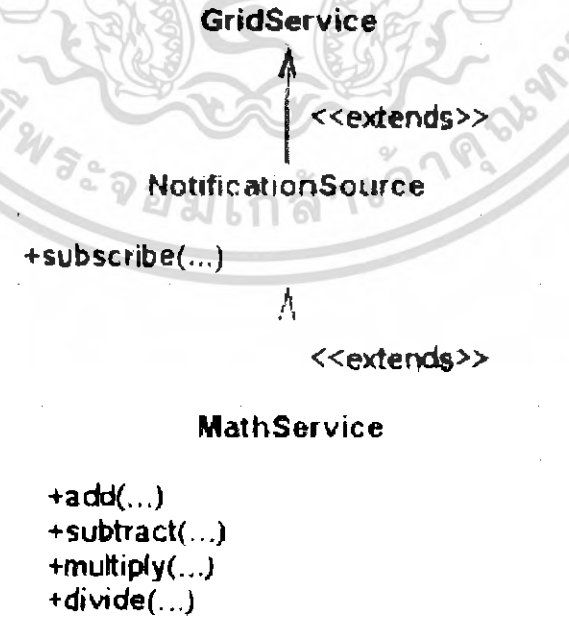
- portType Extension

  เรียกอินเตอร์เฟสของ เว็บเซอร์วิส ว่า portType ซึ่งโดยทั่วไปมักจะมีเพียง 1 portType ต่อ 1 เว็บเซอร์วิส แต่ใน กริดเซอร์วิส สามารถมีได้หลายๆ portType ต่อ 1 เซอร์วิส ซึ่ง portType ที่เพิ่มขึ้นมานั้นต้องขยายมาจาก portType ที่มีอยู่ก่อน



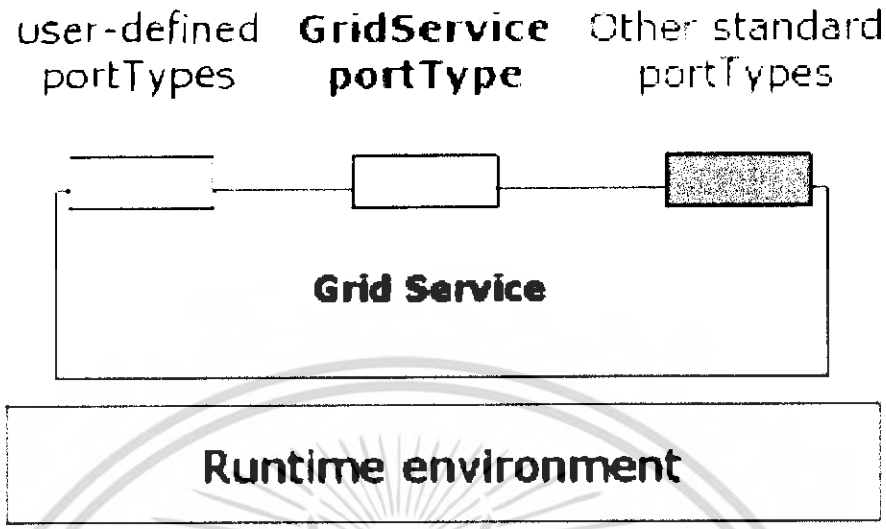
รูปที่ 6-7 แสดงการขยาย portType ของกริดเซอร์วิส

OGSI ออกมาตรฐาน portType อื่นๆที่สามารถขยายได้โดยการเพิ่มฟังก์ชั่น เข้าไปใน กริดเซอร์วิส



รูปที่ 6-8 แสดงการเพิ่มฟังก์ชั่นเข้าไปในกริดเซอร์วิส

โดยทั่วไปแล้ว จะพบว่า กริดเซอร์วิส มี portType อยู่ 3 อย่าง



รูปที่ 6-9 แสดง portType ของกริดเซอร์วิส

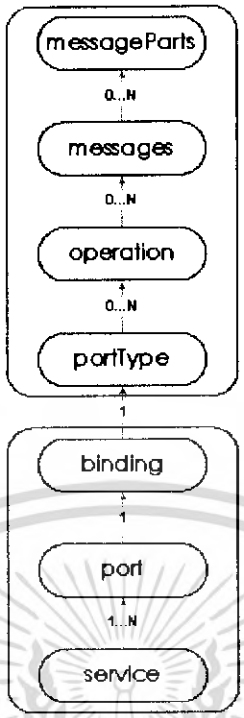
- GSH & GSR

  กริดเซอร์วิส ถูกระบุตำแหน่งด้วย URI เช่นเดียวกับ เว็บเซอร์วิส แต่เรียก กริดเซอร์วิส URI ว่า Grid Service Handle (GSH) เช่นเดียวกับ เว็บเซอร์วิส ที่ต้องมี WSDL ใช้บอกวิธีติดต่อกับ เว็บเซอร์วิส นั้นๆ กริดเซอร์วิส ก็ต้องมี Grid Service Reference (GSR)

6.3 ความแตกต่างของกริดเซอร์วิสและเว็บเซอร์วิส

6.3.1 Web Services Description Language (WSDL)

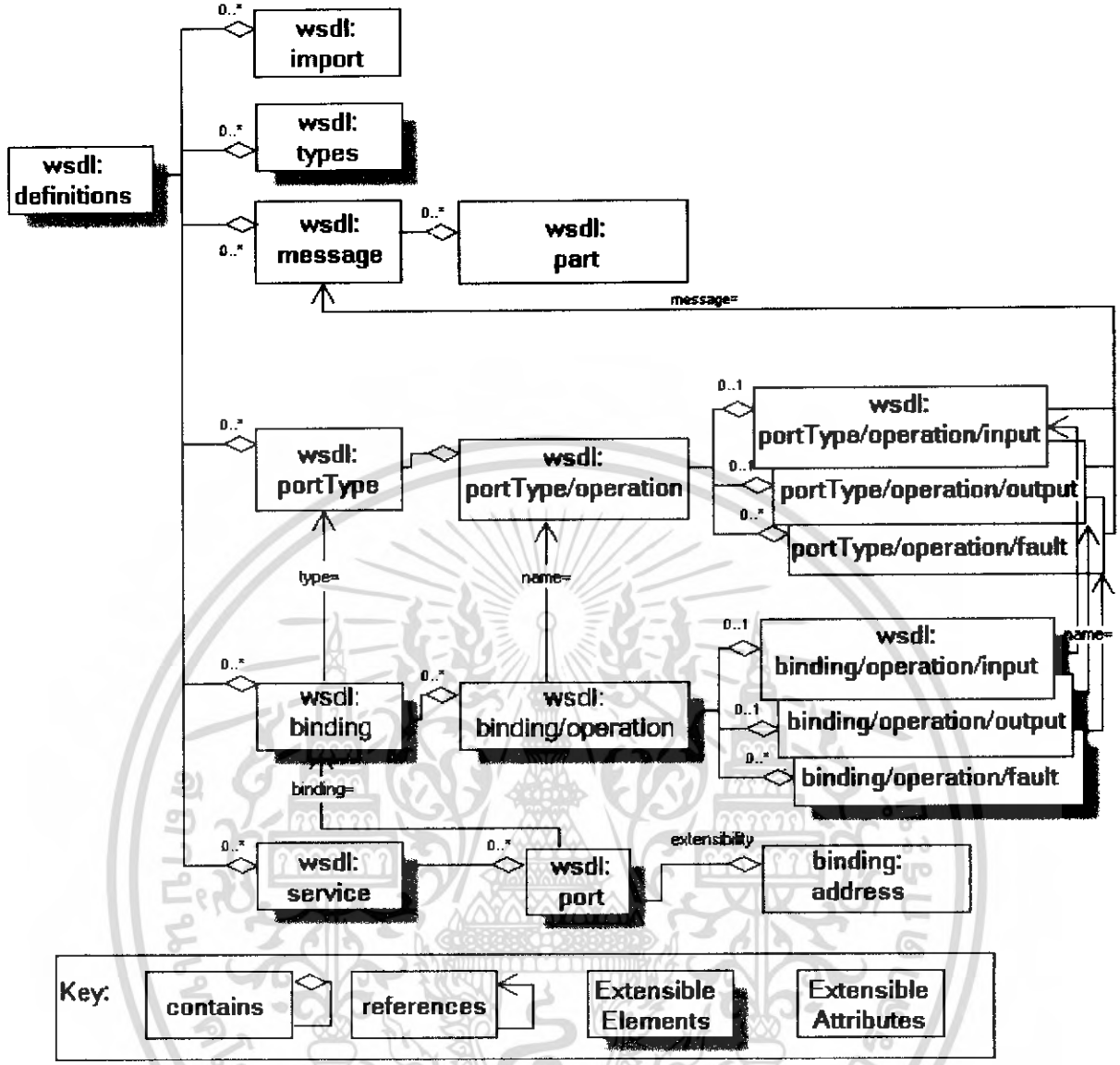
WSDL เป็น XML document เพื่ออธิบายข้อตกลงระหว่างเว็บเซอร์วิสและผู้บริโภคเกี่ยวกับรูปแบบของข้อความว่าสามารถเปลี่ยนเป็นรูปแบบใดได้บ้างที่เซอร์วิสจะรองรับ จากรูป 6-10 เป็นโครงร่างและความสัมพันธ์ของโครงสร้างในส่วนหลักของ WSDL document



รูปที่ 6-10 โครงสร้างเอกสาร WSDL

องค์ประกอบของ document ถูกแบ่งออกเป็น 2 ส่วน:

- ส่วนบน ('message', 'portType') เป็นส่วนสำคัญของ service interface, operations, parameters ซึ่งหมายความว่าผู้บริโภคสามารถทำอะไรได้บ้าง
- ส่วนล่าง ระบุ binding(s) ของวิธีการทำให้ message formats เป็นรูปธรรมขึ้นมา, โปรโตคอล และ ที่อยู่ปลายทางที่สามารถเรียกใช้เซอร์วิสได้



รูปที่ 6-11 ค่าต่างๆของ WSDL

จากรูป 6-11 อธิบายโครงสร้างของ WSDL document ละเอียดขึ้น เริ่มจาก ส่วนประกอบ XML ที่เรียกว่า wsdl:definitions อาจประกอบด้วยส่วนย่อยๆ เช่น wsdl:import และ wsdl:types

หัวใจของ WSDL คือ portType ซึ่งระบุกลุ่ม input, output และ fault message ที่เซอร์วิส เตรียมรับมือไว้แล้ว บางครั้งก็เทียบง่ายๆได้ว่า portType เป็น JAVA interface หรือ C++ class ส่วน message ใน portType อาจประกอบด้วยหลายๆส่วนซึ่งแต่ละส่วนอาจต่างชนิดกัน หรืออาจเป็น input และ output parameters ที่แมสเซสรวมไว้ด้วยกัน

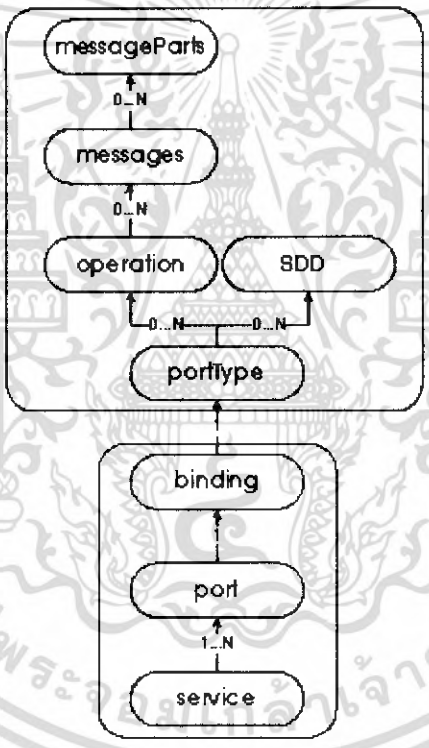
ชนิดของแมสเซสถูกระบุด้วยชนิดของส่วนประกอบใน wsdl:definition ซึ่งสามารถขยายออกไปได้โดยการเข้าไปสร้างส่วนประกอบย่อยเพื่อสร้างชนิดของข้อมูล โดยชนิดโดยปกติใน WSDL คือ

XML Schema

ส่วนสำคัญอื่นๆของ WSDL คือการนำมาผูกกัน (binding) เป็นการสร้างแมสเซสซึ่งก็คือ data encoding, messaging protocol, และ underlying communication protocol ส่วนประกอบ XML ที่ใช้จะเป็น operations ที่ใช้ชื่ออ้างอิงถึงแมสเซสต่างๆ ลักษณะสำคัญของ WSDL คือความจุที่สามารถขยายได้โดยการเพิ่มส่วนประกอบย่อยๆเข้าไป WSDL ยอมให้บรรยาย data encoding, message และ communication protocol ใดๆได้ ประโยชน์ของการบรรยายส่วนต่างๆขึ้นกับระบบไคล์เอนท์และเซิร์ฟเวอร์สามารถแปล WSDL descriptions เพื่อเข้าและถอดรหัสแมสเซส

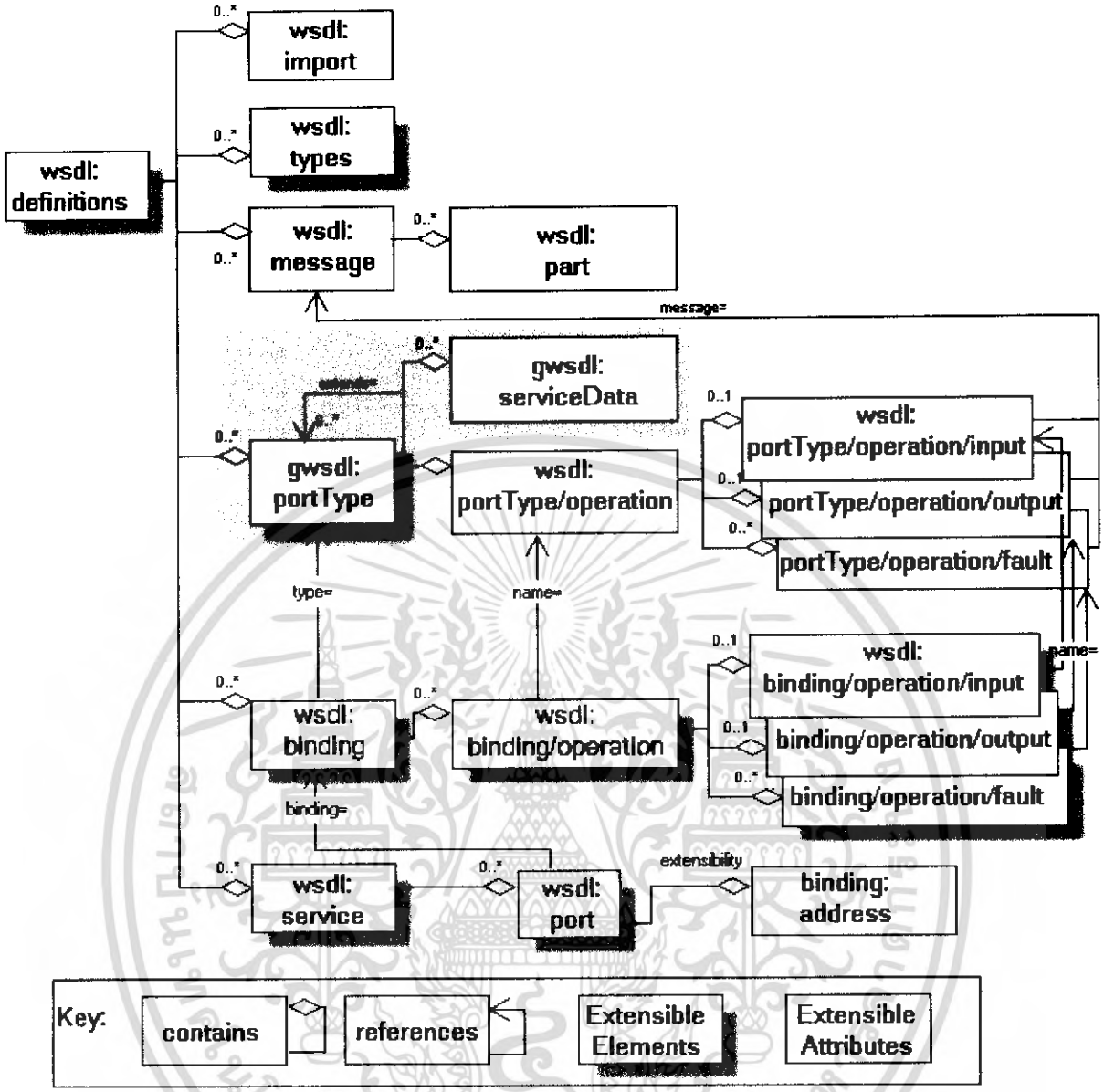
6.3.2 Grid Services Description Language (GWSDL)

OGSI เพิ่ม features ของกริดเซอร์วิสเข้าไปในเว็บเซอร์วิส โดยทำ WSDL portType element ซ้ำโดยกริดเซอร์วิส จะบรรยายในรูปแบบของ WSDL ที่เพิ่มขึ้น ที่เรียกว่า GWSDL รูป 6-12 แสดงโครงสร้างหลักของ GWSDL ซึ่งคล้ายกับโครงสร้างของ WSDL



รูปที่ 6-12 โครงสร้างเอกสาร GWSDL

ข้อแตกต่างระหว่าง GWSDL กับ WSDL แสดงอยู่ในรูปต่อไปนี้



รูปที่ 6-13 Markup language for Grid Services

- ใน OGSI, portType หนึ่งๆ ถูกสร้างโดยการอ้างถึง portType หรือ portTypes ที่มีอยู่กับ attribute ที่เพิ่มขึ้น และการกำหนดความหมายเพิ่มเข้าไปเท่าที่ต้องการ
- ขณะที่ WSDL portType มีแค่ opcrations, OGSI portType ก็มี Service Data Declarations (SDDs) ซึ่งช่วยบรรยาย identity และ interfaces ของ service รวมไปถึงสถานะที่ไคล์เอนท์สามารถดูได้

6.3.3 ตัวอย่างโปรแกรม 'Counter'

จะเปรียบเทียบให้เห็นความแตกต่างระหว่างกริดเซอร์วิส และ เว็บเซอร์วิสโดยอาศัยโปรแกรม

Counter ช่วยในการเปรียบเทียบ

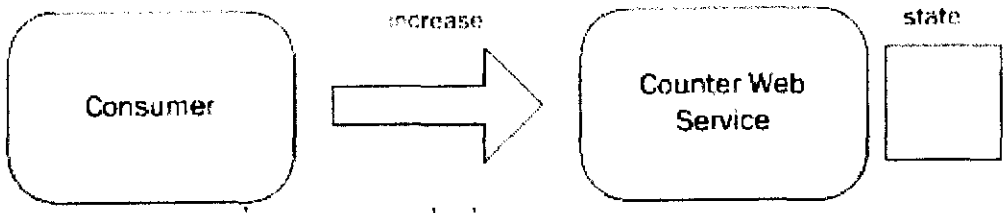
6.3.3.1 Counter Web Service

WSDL สำหรับ Counter Service ทั่วไปที่มี 2 operations (increase และ getValue) แสดงดังรูป 6-14

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:counter="http://www.gridforum.org/namespaces/
2003/05/ogsiprimer/counter/webservice"
        targetNamespace="http://www.gridforum.org/namespaces/
2003/05/ogsiprimer/counter/webservice">
  <wsdl:types>
    <xs:schema />
  </wsdl:types>
  <wsdl:message name="increaseMsg">
    <wsdl:part name="value" type="xs:positiveInteger" />
  </wsdl:message>
  <wsdl:message name="getValueMsg">
    <wsdl:part name="value" type="xs:positiveInteger" />
  </wsdl:message>
  <wsdl:portType name="counterPortType">
    <wsdl:operation name="increase">
      <wsdl:input message="increaseMsg" />
    </wsdl:operation>
    <wsdl:operation name="getValue">
      <wsdl:output message="getValueMsg" />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

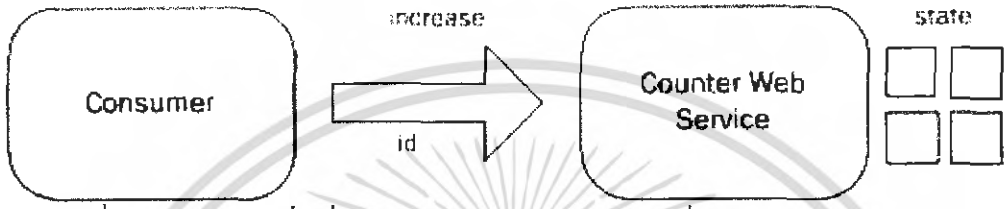
รูปที่ 6-14 เอกสาร WSDL Counter Web Service

ในโลกของ Web Services, Counter Web Service คือ agent ที่คอยแสดงผลเป็น WSDL และเป็นตัวรับ operation requests



รูปที่ 6-15 การใช้คำสั่งเพิ่มค่าใน Counter Web Service

ถ้าทำ Counter Web Service **ให้เป็น** multiple counter **ต้องมี** id **เพิ่มเข้ามาเพื่อใช้ระบุ** counter **ที่กำลังถูกใช้งาน ตามที่แสดงในรูป** 6-16



รูปที่ 6-16 การใช้คำสั่งเพิ่มค่าใน Counter Web Service **ที่รองรับหลายๆ** Counter

```
<wsdl:definitions xmlns:wsdl="http:  schemas.xmlsoap.org wsdl "
        xmlns:wsdl="http:  schemas.xmlsoap.org wsdl"
        xmlns:xs="http:  www.w3.org 2001 XMLSchema"
        xmlns:counter="http:  www.gridforum.org namespaces
2003 05 ogsiprimer counter basic "
        targetNamespace="http:  www.gridforum.org namespaces
2003 05 ogsiprimer counter basic ">
  <wsdl:types>
    <xs:schema >
  < wsdl:types>
  <wsdl:message name="increaseMsg">
    <wsdl:part name="counterId" type="xs:positiveInteger" >
    <wsdl:part name="value" type="xs:positiveInteger" >
  < wsdl:message>
  <wsdl:message name="getValueRequestMsg">
    <wsdl:part name="counterId" type="xs:positiveInteger" >
  < wsdl:message>
  <wsdl:message name="getValueResponseMsg">
    <wsdl:part name="value" type="xs:positiveInteger" >
  < wsdl:message>
```

```
<wsdl:portType name="counterPortType">
  <wsdl:operation name="increase">
    <wsdl:input message="increaseMsg" >
  < wsdl:operation>
  <wsdl:operation name="getValue">
    <wsdl:input  message="getValueRequestMsg" >
    <wsdl:output message="getValueResponseMsg" >
  < wsdl:operation>
< wsdl:portType>
< wsdl:definitions>
```

รูปที่ 6-17 เวอร์ชั่นแก้ไขของ Counter Web Services WSDL ที่รองรับหลายๆ counters

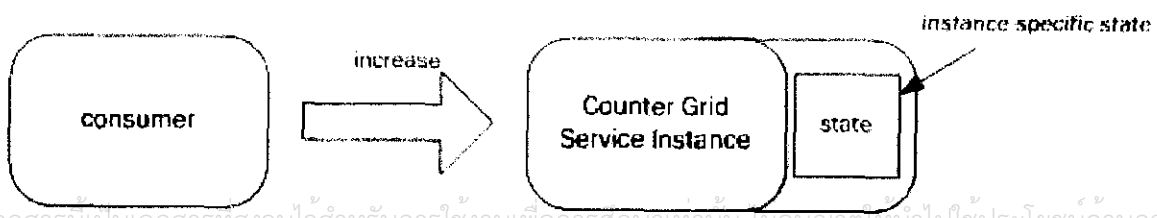
6.3.3.2 Counter Grid Service

ใน OGSI-terms. กริดเซอร์วิส ใช้อ้างอิงถึงแม่แบบอย่างเป็นทางการ เป็นข้อตกลงที่ว่า interface ที่แสดงเป็น GWSDL ต้องมีกริดเซอร์วิสอินสแตนซ์ติดมาด้วย แสดงถึงลักษณะส่วนประกอบของแอพพลิเคชั่นของกริด (ไม่นับรวมถึง agent ที่สามารถรับ operation request ได้) หมายถึง กริดเซอร์วิสอินสแตนซ์ เป็นตัวรับ logical ของ operation requests

ในมาตรฐานของ OGSI, จำเป็นต้องมี กริดเซอร์วิสอินสแตนซ์ ที่กำหนด Grid Service Handle (GSH) ซึ่งถูกระบุขึ้นมาเป็นพิเศษ และมีการลงทะเบียน แล้วจึงให้ กริดเซอร์วิสอินสแตนซ์ อื่นๆหาเจอได้ ขณะที่ GSH มีลักษณะพิเศษ เพื่อสร้าง Grid Service Reference (GSR) ขึ้นมาเพื่อให้ได้ endpoint-related information ซึ่งจำเป็นต้องใช้ในการเข้าถึงกริดเซอร์วิสอินสแตนซ์ที่ระบุไว้

ครั้งหนึ่ง ผู้ใช้สามารถใช้งาน GSR เพื่อทำให้กริดเซอร์วิสอินสแตนซ์ทำงานได้เหมือนกับ Web Service counterpart อย่างไรก็ตาม ลักษณะโดยเฉพาะของ OGSI ของกริดเซอร์วิสอินสแตนซ์นั้นสามารถใช้ประโยชน์ได้มากกว่านี้

Counter GSI (Counter Grid Service Instance) ดังแสดงในรูป 3-18 เหมือนกับ Counter Web Service ที่ได้แสดงไปแล้วข้างต้น ผู้ใช้งานเซอร์วิสหลายคนสามารถขอคำสั่งบน Counter GSI เดียวกันได้ เพื่อเข้าถึงหรือแก้ไขสเตทของ counter นี่เป็นข้อแตกต่างอย่างหนึ่งจาก Web Service ที่ต้องติดต่อเป็น logic ไปยัง อินสแตนซ์



รูปที่ 6-18 การใช้คำสั่งเพิ่มค่า Counter Grid Service Instance

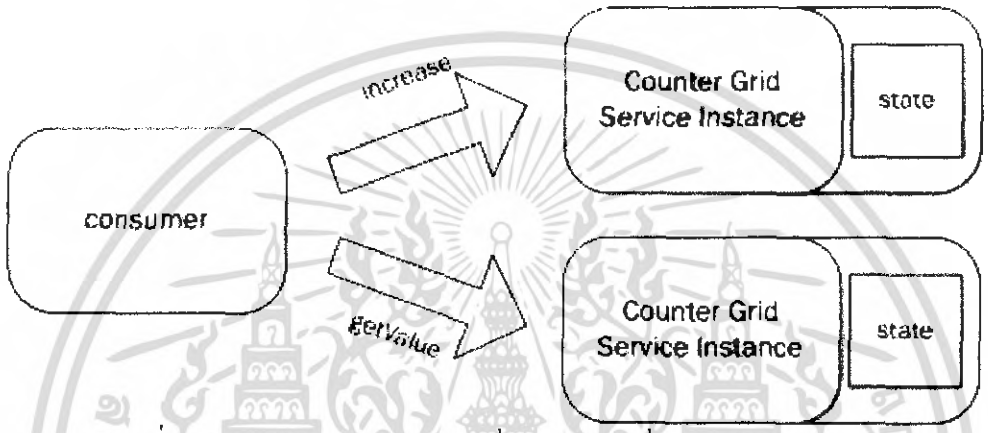
ด้วยความแตกต่างจาก Counter Web Service ทำให้ Counter Grid Service Instance ต้องมีคุณสมบัติในการจัดการไลฟ์ไทม์ ซึ่งใช้จัดการกับวัฏจักรชีวิตของ counter

GWSDL interface ของ Counter Grid Service ดูคล้ายกับ เว็บเซอร์วิส นอกจากเปลี่ยน namespace อย่างเห็นได้ชัดแล้ว ข้อแตกต่างอื่นๆดังรูปด้านล่าง แสดงถึง GWSDL processors ที่ counterPortType ขยายการใช้งานออกไปด้วย ogsi:GridService portType

```
<wsdl:definitions xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:counter="http://www.gridforum.org/namespaces/
2003/05/ogsiprimer/counter/basic"
        targetNamespace="http://www.gridforum.org/namespaces/
2003/05/ogsiprimer/counter/basic">
  <wsdl:types>
    <xs:schema />
  </wsdl:types>
  <wsdl:message name="increaseMsg">
    <wsdl:part name="value" type="xs:positiveInteger" />
  </wsdl:message>
  <wsdl:message name="getValueMsg">
    <wsdl:part name="value" type="xs:positiveInteger" />
  </wsdl:message>
  <ogsi:portType name="counterPortType" extends="ogsi:GridService">
    <wsdl:operation name="increase">
      <wsdl:input message="increaseMsg" />
    </wsdl:operation>
    <wsdl:operation name="getValue">
      <wsdl:output message="getValueMsg" />
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

รูปที่ 6-19 เอกสาร GWSDL ของ Counter Grid Service

แน่นอนว่าความสัมพันธ์ระหว่างกริดเซอร์วิสอินสแตนซ์และสเตทของมันยอมให้เราสำเนาการใช้งานของ Multiple-Counter Web Service โดยไม่ส่งผลถึง interface ทั้งหมดนี้ กล่าวคือ กริดเซอร์วิสอินสแตนซ์เป็นรูปแบบใหม่จาก Web Service ดังรูป 6-18 เวลาต้องการใช้งาน counter หลายครั้ง ค่าเดิมจะยังคงอยู่ใน interface เดิม อย่างไรก็ตาม OGSI ก็สนับสนุนให้ กริดเซอร์วิสอินสแตนซ์ แสดงสเตทได้ชั่วคราว ดังนั้น จึงเป็นธรรมดาที่จะแสดงแต่ละ counter ด้วยแต่ละ กริดเซอร์วิสอินสแตนซ์ กล่าวคือ มีความสัมพันธ์แบบ 1 ต่อ 1 ระหว่างทรัพยากรหนึ่ง(ในที่นี้คือ counter) กับกริดเซอร์วิสอินสแตนซ์



รูปที่ 6-20 การใช้ 2 Counter GSI ที่แตกต่างกันที่ใช้ในแต่ละ counter

นอกจากการจัดการสเตทแล้ว กริดเซอร์วิสอินสแตนซ์ สามารถเข้าถึงข้อมูลผ่านโครงสร้างที่เรียกว่า Service Data Elements (SDEs) ที่ประกาศใน interface ของกริดเซอร์วิสผ่านทาง คำสั่งที่ระบุโดย GridService portType ของ OGSI

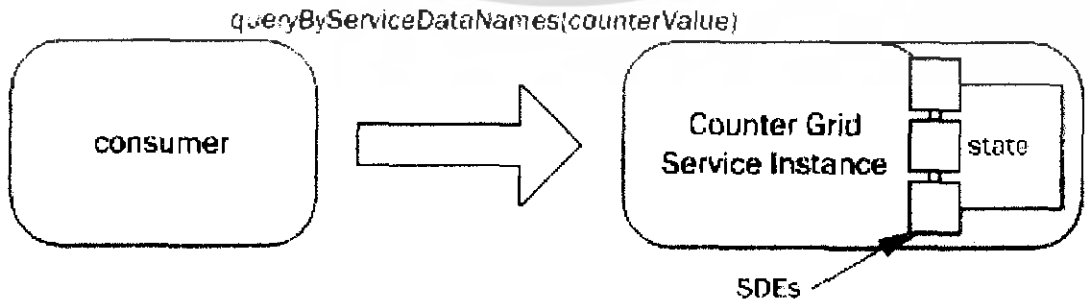
Counter Grid Service อาจเผยค่าของ counter ผ่าน SDE โดยไม่ต้องพึ่งคำสั่ง getValue รูป 3-21 แสดง GWSDL ของ Counter Grid Service ที่มี SDE

```
<wsdl:definitions xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
        xmlns:sd="http://www.gridforum.org/namespaces/
2003/03/serviceData"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:counter="http://www.gridforum.org/namespaces/
2003/05/ogsiprimer/counter/basic"
        targetNamespace="http://www.gridforum.org/namespaces/
2003/05/ogsiprimer/counter/basic">
  <wsdl:types>
```

```
    <xs:schema>
  < wsdl:types>
  <wsdl:message name="increaseMsg">
    <wsdl:part name="value" type="xs:positiveInteger" >
  < wsdl:message>
    <wsdl:message name="getValueMsg">
    <wsdl:part name="value" type="xs:positiveInteger" >
  < wsdl:message>
  <ogsi:portType name="counterPortType" extends="ogsi:GridService">
    <wsdl:operation name="increase">
      <wsdl:input message="increaseMsg" >
    < wsdl:operation>
    <sd:serviceData name="counterValue"
                type="xs:positiveInteger"
                minOccurs="1"
                maxOccurs="1"
                mutability="mutable">
      <sd:documentation>
        The value of the counter.
      < sd:documentation>
    < sd:serviceData>
  <ogsi:portType>
< wsdl:definitions>
```

รูปที่ 6-21 GWSDL สำหรับ Counter Grid Service ที่ใช้ค่า SDE

ตอนนี้จึงเป็นไปได้ที่จะเข้าถึง counterValue SDE ผ่าน operations ที่กำหนดโดย GridService portType (รูป 6-22)



รูปที่ 6-22 การร้องขอค่า counter **จากลูกค้าผ่าน** counterValue SDE

บทที่ 7

มาตรฐาน OGSA & OGSI

ในการพัฒนาระบบกริดนั้นได้มีผู้พัฒนามากมาย จากหลายๆที่ ซึ่งแต่ละที่แม้จะมีแนวความคิดที่เหมือนกัน แต่ถ้าจะนำมาใช้แล้ว จะสามารถใช้งานเข้าด้วยกันได้หรือไม่ ด้วยเหตุนี้เองทำให้ต้องมีการกำหนดมาตรฐานขึ้นมา เพื่อให้สามารถใช้งานระบบกริดร่วมกันได้ขึ้นมา โดยกำหนดออกมาเป็น Open Grid Service Architecture (OGSA)

OGSA เปรียบเสมือนพิมพ์เขียวในการที่จะสร้างมิดเดิ้ลแวร์ขึ้นมา หรือสามารถกล่าวได้ว่า OGSA เป็นโปรโตคอลพื้นฐานที่กำหนดวิธีการการติดต่อ, การส่งข้อมูล, การจัดเตรียมรีซอร์ส, การแชร์รีซอร์ส, การทำตารางเวลา และอื่นๆที่เกี่ยวข้อง โดยอาศัยเทคโนโลยีอยู่ 2 อย่างในการ สร้างมันขึ้นมา คือ เครื่องมือ และ เว็บเซอร์วิส โดยสร้างขึ้นมาเป็น Open Grid Service Infrastructure (OGSI) ที่จะนำมาใช้งานจริง

OGSI จะเป็นวิธีการในการสร้างกริดเซอร์วิส ซึ่งจะหมายถึงการจัดการ การแลกเปลี่ยนข้อมูลระหว่าง ผู้ใช้งานกริด และ เซอร์วิส อื่นๆ หรืออาจพูดได้ว่า เป็น เว็บเซอร์วิสที่จะช่วยเพิ่มความสะดวกให้กับการทำระบบกริด

7.1 Open Grid Service Architecture

ภายในโครงสร้างไอทีที่ใช้ในธุรกิจนั้น จะใช้ ส่วนจัดเตรียมเซอร์วิส (Service Provider) เป็นส่วนช่วยในการเพิ่มประสิทธิภาพให้โครงสร้างของระบบไอทีและการคำนวณแบบกริดจากหลายๆที่ โดยในการคำนวณนั้นจะเกี่ยวข้องกับการสร้าง การจัดการ และ แอพพลิเคชั่น ที่เกิดจากการนำมารวมกันของ รีซอร์ส และ เซอร์วิส (และ ผู้ใช้งาน) ที่เรียกว่า Virtual Organizations ซึ่ง VO นั้นจะเป็นได้ทั้งแบบ เล็ก ใหญ่ มีอายุสั้น อายุยาว ใช้แบบเดี่ยวใช้แบบหลาย และเป็นเนื้อเดียวกัน (homogeneous) หรือ เป็นแตกต่างกัน (heterogeneous)

Open Grid Service Infrastructure นั้นเป็นโครงสร้างที่ถูกกำหนดให้รองรับการสร้าง การดูแลรักษา และให้เป็น แอพพลิเคชั่นของเซอร์วิสที่ดูแลโดย VOs

7.1.1 Service Orientation and Virtualization

เมื่อกล่าวถึง VO จะมองรวมไปที่ รีซอร์สทางกายภาพ นั้นถูกใช้งานร่วมกันอยู่ หรือ อาจจะเป็นเซอร์วิส นั้นจะได้รับความช่วยเหลือจาก resource เหล่านั้น โดยใน OGSA ได้ให้คำจำกัดความของเซอร์วิสว่ารีซอร์สที่ใช้ในการคำนวณ,รีซอร์สที่เป็นอุปกรณ์เก็บข้อมูล,ระบบเครือข่าย,โปรแกรม, ฐานข้อมูลและทุกสิ่งที่คล้ายๆกับที่กล่าวมาว่าเป็นเซอร์วิส

Service-oriented จะทำให้สามารถกำหนดสิ่งที่ต้องการทั้งหมดเพื่อกำหนดมาตรฐานใน กลไกการทำงาน (mechanisms),ลักษณะการใช้งานแบบ local หรือ remote (local/remote transparency), การดัดแปลงให้เหมาะสมกับ OS services (adaptation to local OS services) และ ทำเซอร์วิสให้อยู่ในรูปแบบเดียวกัน (uniform service semantics) ซึ่ง Service-oriented นั้น จะทำให้ Virtualization แก้ไขได้ง่าย ซึ่งเป็นความสามารถที่ encapsulation ภายใต้การ implement ในรูปแบบต่างๆ โดยที่ Virtualization ของกริดเซอร์วิสจะทำให้สามารถ map เซอร์วิสพื้นฐานได้สะดวกบนแต่ละ แพลทฟอร์ม

Virtualization จะง่าย ถ้า ฟังก์ชั่นของเซอร์วิสสามารถแสดงได้ในรูปแบบมาตรฐาน ดังนั้น ในการ สร้างเซอร์วิส ควรจะถูก invoke ด้วยวิธีเดียวกัน โดย WSDL ถูกนำมาใช้ในกรณีนี้ โดย WSDL จะให้ทำ multiple bindings สำหรับรูปแบบเดี่ยวได้ โดยรวมถึงการกระจายโปรโตคอลสื่อสาร (เช่น HTTP) ได้ดี เช่นเดียวกับ locally optimized binding (เช่น local IPC) สำหรับการตอบโต้กันระหว่างการร้องขอและ กระบวนการต่างๆของเซอร์วิสบนเครื่องเดียวกัน

7.1.2 Service Semantics: The Grid Services

ในการที่จะทำให้เกิดมาตรฐานในการตอบโต้ของ เซอร์วิสได้นั้น จะต้องมีเซอร์วิสที่ต่างกันและใช้ในการแจ้งความผิดพลาดขึ้น ซึ่ง OGSA จะเป็นตัวที่กำหนด กริดเซอร์วิส ขึ้น ซึ่ง กริดเซอร์วิส นั้นคือเว็บเซอร์วิสซึ่งจัดเตรียมกลุ่มของรูปแบบที่กำหนดไว้และทำตามแบบแผนโดยเฉพาะโดยที่รูปแบบนี้จะทำให้ ค้นหาที่อยู่, สร้างเซอร์วิสแบบไดนามิค, จัดการไลฟ์ไทม์, ตรวจสอบ และควบคุมได้ง่าย

รูปแบบและแบบแผนที่กำหนด กริดเซอร์วิส นั้นย่อมต้องเกี่ยวข้องกัน โดยเฉพาะส่วนที่เกี่ยวข้องกันกับการจัดการ เซอร์วิสอินสแตนซ์แบบชั่วคราว ยกตัวอย่างเช่น เซอร์วิสอินสแตนซ์ อาจจะเป็นส่วนที่ คิวรี่ฐานข้อมูล, ทำเหมืองข้อมูล, จองแบนด์วิธของระบบเครือข่าย, ขนย้ายข้อมูล หรือรองรับความสามารถในการประมวลผล เป็นต้น โดยแบบชั่วคราวนั้นจะเป็นส่วนที่แสดงว่า เซอร์วิสจะถูก จัดการ, มีชื่อ, ค้นหา และใช้งานได้อย่างไร

7.1.2.1 Upgradeability Conventions and Transport Protocols

เซอร์วิสภายในระบบนี้ต้องมีความสามารถที่จะอัพเกรดได้ โดยที่ไม่ทำให้ไคล์เอนท์ที่ทำงานอยู่เกิดปัญหาขึ้น เช่น การ อัพเกรดเครื่องที่ใช้ทำระบบนั้น อาจทำให้โปรโตคอลของระบบเครือข่ายเปลี่ยน โดย OGSA ได้กำหนดโครงสร้างที่จะทำการรีเฟรชให้ไคล์เอนท์รู้เกี่ยวกับเซอร์วิส ที่มีการอัพเกรดขึ้น เช่น บอกว่างานใดที่รองรับ หรือ บอกว่า โปรโตคอลเครือข่ายใดที่รองรับกับเซอร์วิสนั้นๆ เป็นต้น โดยที่เซอร์วิสนั้น จะต้องมี 2 คุณสมบัติที่สำคัญ ได้แก่

- Reliable service invocation คือ เซอร์วิสจะตอบโต้กับเซอร์วิสอื่นๆโดยการแลกเปลี่ยนข้อความในระบบแบบกระจายนั้น จะต้องมีความแน่นอนว่า ข้อความนั้นส่งไปถึงที่หมายได้จริงหรือไม่

- Authentication แต่ละเซอร์วิสนั้นจะต้องเป็นไปตามนโยบายที่กำหนดไว้ เช่น มีการตรวจสอบไคล์เอนท์หรือตรวจสอบเซอร์วิสอินสแตนซ์ก่อนใช้งาน

7.1.2.2 Standard Interfaces

Interfaces ที่ใช้กำหนดกริดเซอร์วิสนั้นได้ถูกแสดงไว้ดังตารางด้านล่าง

PortType	Operation	Description
GridService	FindServiceData	Query a variety of information about the Grid service instance, including basic introspection information (handle, reference, primary key, home handleMap; terms to be defined), richer per-interface information, and service-specific information (e.g., service instances known to a registry). Extensible support for various query languages.
	SetTerminationTime	Set (and get) termination time for Grid service instance
	Destroy	Terminate Grid service instance
Notification-Source	SubscribeTo-NotificationTopic	Subscribe to notifications of service-related events, based on message type and interest statement. Allows for delivery via third party messaging services.
Notification-Sink	DeliverNotification	Carry out asynchronous delivery of notification messages
Registry	RegisterService	Conduct soft-state registration of Grid service handles
	UnregisterService	Deregister a Grid service handle
Factory	CreateService	Create new Grid service instance
HandleMap	FindByHandle	Return Grid Service Reference currently associated with supplied Grid Service Handle

ตารางที่ 7-1 อินเตอร์เฟสมาตรฐานของกริดเซอร์วิส

แอพพลิเคชั่นนั้นจะต้องการกลไกที่จะใช้ค้นหาเซอร์วิสที่ใช้งานได้ และสำหรับตัดสินใจเลือกลักษณะพิเศษของเซอร์วิส ที่จะใช้งานที่จะทำงานได้ตามที่ร้องขอ ตามนี้

- มาตรฐานที่ใช้แสดงข้อมูลของเซอร์วิส เป็นการแสดงข้อมูลเกี่ยวกับ กริดเซอร์วิสอินสแตนซ์ (Grid Services Instances) ซึ่ง มีโครงสร้างเป็นกลุ่มของชื่อและอยู่ในรูปแบบ XML ซึ่งเรียกว่า Service data elements ซึ่ง encapsulate เป็นรูปแบบมาตรฐานไว้

- มาตรฐานของคำสั่งต่างๆ เช่น FindServiceData ที่ใช้สำหรับนำข้อมูลจาก กริดเซอร์วิสอินสแตนซ์ กลับมาใช้งาน

- มาตรฐานของการลงทะเบียนข่าวสารเกี่ยวกับ กริดเซอร์วิส ด้วย Registry Services และทำการ map จาก handle ให้เป็น reference จะกล่าวในหัวข้อต่อๆไป

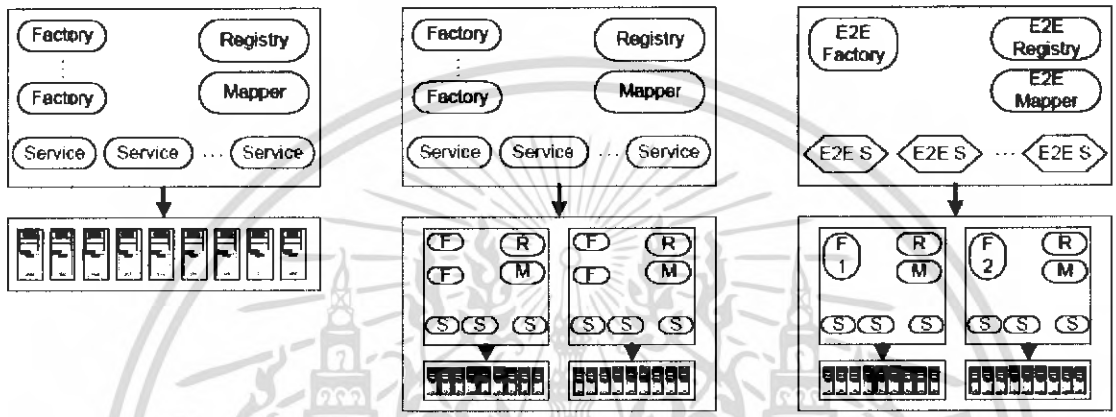
7.1.3 The Role of Hosting Environment

OGSA ได้กำหนดเกี่ยวกับ กริดเซอร์วิสอินสแตนซ์ ไว้ กล่าวคือ กำหนดว่า สร้างขึ้นมาอย่างไร ให้ชื่อได้อย่างไร มีวัฏจักรชีวิตนานเท่าไหร่ ติดต่อกับส่วนใดบ้าง เป็นต้น อย่างไรก็ตาม OGSA บอกในส่วนที่เป็นพฤติกรรมเบื้องต้น มันไม่ได้บอกว่าเซอร์วิสทำอะไร และไม่ได้บอกว่าทำเซอร์วิสขึ้นมา

อย่างไร เพราะ OGSA ไม่ได้บ่งบอกเกี่ยวกับ Programming Model, Programming Language, Implementation Tool หรือ Execution Environment เลย แต่ในปัจจุบันก็มีการใช้ภาษาหลากหลายรูปแบบในการสร้างเซอร์วิสเช่น C, C++, Java or Fortran

7.1.4 Using OGSA Mechanisms to Build VO Structures

แอพพลิเคชั่นและผู้ใช้งาน ต้องสามารถที่จะสร้างเซอร์วิสชั่วคราว และ ค้นหาและเลือกใช้ตามคุณสมบัติได้ ใน OGSA จะมี Factory, Registry, GridService and HandleMap ที่เป็นอินเตอร์เฟสที่รองรับการสร้างเซอร์วิสอินสแตนซ์ชั่วคราวและ ค้นหาและเลือกเซอร์วิสอินสแตนซ์ที่เกี่ยวข้องกับ VO ได้



รูปที่ 7-1 ตัวอย่าง 3 รูปแบบที่ใช้งาน

1. Simple Hosting Environment เป็น set ของ รีซอร์ส ที่ตั้งใน single administrative domain และ รองรับการจัดการเซอร์วิสโดยแต่ละ factory จะถูกบันทึกในรูปแบบ registry เพื่อให้ไคล์เอนท์สามารถค้นพบ factories ที่ใช้งานได้ เมื่อมี factory ที่ถูกไคล์เอนท์ร้องขอเพื่อสร้าง กริดเซอร์วิสอินสแตนซ์ แล้ว factory จะ invoke host-environment ให้สร้างอินสแตนซ์ใหม่ขึ้น และใส่ค่า handle และ register instance นั้น และสร้าง handle ที่ใช้งานได้ลงใน handleMap โดยลักษณะแบบนี้กล่าวได้อีกอย่างว่าเป็น Local Operations

2. Virtual Hosting Environment เช่นเดียวกับแบบแรก แต่จะเป็นการกระจายไปสู่หลายๆ hosting environment โดยที่ client ที่จะใช้งาน จะต้อง register ใน VO ก่อน เพื่อใช้ในการค้นหาและเรียกใช้งาน เป็นการจำลองการทำงานของ 2 environment ที่ต่างกัน แต่ใช้ทำงานร่วมกัน

3. Collective Services เปรียบเสมือนกับการสร้าง Virtual Hosting Environment หลายๆอันโดยที่แต่ละส่วนนั้นจะเป็นเซอร์วิสอินสแตนซ์ระดับสูง (High Level Service Instance) ต่างกับแบบที่ 2 ที่จะเป็นเพียงเซอร์วิสอินสแตนซ์แบบธรรมดาโดยเซอร์วิสอินสแตนซ์ระดับสูงนั้นจะเปรียบเสมือนเซอร์วิสอินสแตนซ์หลายๆอันที่รวมกันอยู่

7.2 รายละเอียดทางเทคนิคของ OGSA

ในส่วนนี้จะอธิบายรายละเอียดเกี่ยวกับ กริดเซอร์วิส แบบลึกๆ และความเกี่ยวข้องกับรูปแบบตามข้อตกลง

7.2.1 OGSA Service Model

ในส่วนของ OGSA นั้นกล่าวได้ว่าเป็นส่วนที่กล่าวถึงเซอร์วิสซึ่งเป็น entity ทางเครือข่ายที่จัดเตรียมความสามารถในการแลกเปลี่ยนข้อความ อันได้แก่รีซอร์สที่ใช้ในการคำนวณ,รีซอร์สทางกายภาพ, ระบบเครือข่าย, โปรแกรม, ฐานข้อมูล และ เซอร์วิสทั้งหมด โดยที่ OGSA จะแสดงทุกๆอย่างที่กล่าวมาให้เป็น กริดเซอร์วิส

กริดเซอร์วิส เป็นการแสดงลักษณะที่มันสามารถทำได้ ซึ่งจะ implement ออกมาเป็น 1 หรือหลายๆอินเตอร์เฟส โดยที่แต่ละอินเตอร์เฟสนั้นจะกำหนดกลุ่มของกระบวนการที่จะ invoke โดยการแลกเปลี่ยนการกำหนดลำดับของข้อความ ซึ่ง กริดเซอร์วิส จะตอบโต้กันผ่านทาง portTypes ของ WSDL ซึ่งใน set ของแต่ละ portTypes นั้นจะถูกรองรับด้วย กริดเซอร์วิส

กริดเซอร์วิส จะมีการดูแลสเตทภายในของมัน สำหรับการจัดการไลฟ์ไทม์ของเซอร์วิส ซึ่งช่วงที่คงอยู่ของสเตทนั้นจะแบ่งแยกออกเป็น 1 อินสแตนซ์ของเซอร์วิสซึ่งจะใช้กริดเซอร์วิสอินสแตนซ์เป็นส่วนอ้างอิงถึงกริดเซอร์วิส

การผูกโปรโตคอลที่เกี่ยวข้องกับเซอร์วิสที่ส่งออกไปนั้น เช่น เซอร์วิสมีการตอบโต้กับอีกเซอร์วิส โดยการแลกเปลี่ยนแมสเซสทั้งที่ไม่ได้ต้องการให้ในระบบการคำนวณแบบกระจายนั้น จะถือว่าเป็นการล้มเหลว อย่างไรก็ตามส่วนนั้นไม่สามารถรับรองได้ว่าได้ถูกส่งจริง แต่จะใช้การคงอยู่ของสเตทภายในของมันเป็นตัวที่รับรองว่าเซอร์วิสได้รับข้อความหรือยัง แม้ว่าจะต้องใช้การส่งใหม่อีกครั้งถ้ายังไม่ได้รับก็ตาม

OGSA เซอร์วิสจะถูกสร้างหรือถูกทำลายแบบ dynamic ซึ่งบางที เซอร์วิสอาจจะถูกทำลายจริงหรือ ไม่สามารถเข้าถึงระบบได้เนื่องจากเหตุการณ์หลายๆกรณี ไม่ว่าจะเป็นกรณีผลลัพธ์ของระบบผิดพลาด,ระบบปฏิบัติการล่มหรือ ระบบเครือข่ายล่มทำให้ต้องมีอินเตอร์เฟสที่จะกำหนดการจัดการเกี่ยวกับไลฟ์ไทม์

เนื่องจาก กริดเซอร์วิส เป็นแบบ dynamic และ stateful ทำให้มันจะต้องมีทางที่จะแยกการสร้างเซอร์วิสอินสแตนซ์ จากอันอื่นๆที่สามารถสร้างขึ้นมาได้เหมือนกัน ในทุกๆ กริดเซอร์วิสอินสแตนซ์จะต้องมี Global Unique Name ที่เรียกว่า Grid Service Handle (GSH) ที่จะใช้แยก กริดเซอร์วิสอินสแตนซ์ จากอันอื่นๆที่ยังคงอยู่ ไม่ว่าในปัจจุบัน หรือในอนาคตที่กำลังจะมีอันใหม่เพิ่มขึ้นมาอีก (ในกรณีที่ล้มเหลว แล้วมีการสร้างอินสแตนซ์อันเดิม ขึ้นมาใหม่ จะใช้ GSH อันเดิมที่เคยใช้งาน)

ซึ่งกริดเซอร์วิสนั้นจะสามารถอัพเกรด ได้ในระหว่างที่ยังมีชีวิตอยู่ เช่น เมื่อมีการใช้โปรโตคอล เวอร์ชั่นใหม่ หรือ เพิ่มโปรโตคอล อื่นๆเข้าไป ด้วยเหตุนี้ทำให้ GSH จะไม่อยู่บนปรโตคอล หรืออินสแตนส์จะไม่มีข้อมูลที่เจาะจง เช่น ที่อยู่บนเครือข่ายหรือโปรโตคอลที่รองรับ โดยข้อมูลนี้จะถูก encapsulate กับ อินสแตนซ์ อื่นๆที่ใช้ระบุเจาะจงเกี่ยวกับการตอบโต้กับ เซอร์วิสอินสแตนซ์ ที่เรียกว่า

Grid Service Reference (GSR) ที่แตกต่างกับ GSH โดย กริดเซอร์วิสอินสแตนซ์ จะเปลี่ยนได้กระทั่ง Service Lifetime เนื่องจาก GSR จะมีเวลาหมดอายุอย่างตรงๆ หรือแม้กระทั่งจะยกเลิกได้แม้กระทั่งในไลฟ์ไทม์ และ OGSA จะกำหนดการ mapping สำหรับการอัพเดต GSR

อย่างที่บอกว่าใน OGSA นั้นทุกๆอย่าง ถือเป็น กริดเซอร์วิส ซึ่งทำให้จะต้องมี กริดเซอร์วิส ที่ใช้ในการสร้าง กริดเซอร์วิส, handle และ reference ที่ได้กำหนดไว้ใน OGSA model

7.2.2 Creating Transient Services: Factories

OGSA ได้กำหนด class ของ กริดเซอร์วิส ที่ใช้สร้าง กริดเซอร์วิสอินสแตนซ์ ซึ่งเราเรียกว่า Factory ซึ่งใน Factory interface's CreateService จะเป็น operation ที่ใช้สร้างการร้องขอ กริดเซอร์วิส และส่งค่า GSH และค่าเริ่มต้นของ GSR ไปให้เซอร์วิสอินสแตนซ์ ใหม่

ใน Factory interface นั้น ไม่ได้ระบุว่า เซอร์วิสอินสแตนซ์ สร้างขึ้นมาได้อย่างไร เนื่องจาก factory interface จะถูก implement บางส่วนจาก hosting environment (เช่น .NET หรือ J2EE) ที่มีกลวิธีในการสร้าง Service instance ใหม่ๆ ซึ่ง hosting environment จะกำหนดว่า เซอร์วิสจะสร้างได้อย่างไร (เช่น ใช้ภาษาอะไรเขียน เป็นต้น) โดยที่การสร้าง factory ที่มีระดับสูงขึ้นไปนั้น อาจจะใช้การสร้างจากการรวมกันของหลายๆ factory

7.2.3 Service Lifetime Management

ในการจัดการไลฟ์ไทม์ นั้น มีอยู่ 2 กรณี คือ

- ไคล์เอนท์ทราบหรือเป็นตัวตัดสินใจที่จะให้กริดเซอร์วิสนั้นถูกยกเลิกไป ส่วนนี้เป็นส่วนที่จะทำให้ไคล์เอนท์แน่ใจที่จะได้รับข่าวสารว่า เซอร์วิสอินสแตนซ์ นั้นได้ถูกยกเลิกไปแล้ว และ รีซอร์ส ที่ใช้ไปได้กลับคืนมาแล้ว แม้ว่าจะเกิดจากกรณีที่ระบบล้มเหลว (เช่น เกิดจากการล่มของเซิร์ฟเวอร์,เครือข่าย หรือ ไคล์เอนท์) ซึ่งไคล์เอนท์จะต้องทราบสถานะสุดท้ายของ เซอร์วิสอินสแตนซ์ หรือ การร้องขอนั้น ถ้าเป็นในกรณีที่ระบบล่มจะต้องทราบว่ามันจะต้องติดต่อ เซอร์วิส ไหนต่อหลังจากที่มันยกเลิกไปแล้ว โดยที่ทรัพยากรที่เกี่ยวข้องกับเซอร์วิสนั้นๆจะถูกปลดปล่อยหลังจากเวลานั้น และอย่างน้อยไคล์เอนท์จะต้องสามารถขยายไลฟ์ไทม์ได้ด้วย
- Hosting Environment เป็นตัวรับประกันว่ารีซอร์สนั้นเป็นส่วนที่สิ้นเปลือง แม้ว่าจะเกิดจากกรณีการผิดพลาดนอกการควบคุม ก็ตาม ถ้าเวลาที่ควรยกเลิกเซอร์วิสมาถึง hosting environment นั้นจะต้องสามารถที่จะเรียกรีซอร์สที่เกี่ยวข้องคืน

ซึ่งกลวิธีของมันจะแบ่งออกเป็นรอบๆ ตามนี้

- Negotiation an initial lifetime เป็นช่วงที่มีการร้องขอที่จะสร้าง กริดเซอร์วิสอินสแตนซ์ ใหม่ผ่าน factory client จะต้องเป็นตัวแสดงเวลาที่น้อยที่สุดและมากที่สุดที่จะยอมรับได้ในการตั้ง initial lifetime และ factory จะเป็นตัวเลือกค่า initial lifetime นั้นส่งกลับไปให้ไคล์เอนท์

- Request a lifetime extension เป็นช่วงที่ไคล์เอนท์จะร้องขอให้ขยายไลฟ์ไทม์ผ่านข้อความที่ชื่อว่า SetTerminalTime เข้าไปให้กริดเซอร์วิสอินสแตนซ์ซึ่งจะระบุเวลาที่ยอมรับได้ที่น้อยและมากที่สุดไปให้

7.2.4 Managing Handles and Reference

จากที่กล่าวไปในช่วงต้น การตอบของ Factory นั้น จะตอบเป็น GSH และ GSR โดยที่ GSH เป็นการรับรองการอ้างถึงการสร้างเซอร์วิสอินสแตนซ์ และ GSR จะถูกสร้างภายในช่วงเวลาจำกัด และจะเปลี่ยนได้ในระยะเวลาของ service lifetime การที่ GSR มันสามารถเปลี่ยนแปลงได้นั้นอาจทำให้ผิดพลาดได้ ซึ่งจะต้องมีวิธีการที่จะใช้เพียง GSH เพื่อนำมาใช้หาค่า GSR ที่ถูกต้องได้

วิธีการนั้นก็คือ HandleMap ซึ่งเป็นกระบวนการที่จะรับค่า GSH และตอบกลับเป็น GSR ที่ถูกต้อง กระบวนการ map ของ HandleMap นั้น จะเก็บ track ว่า กริดเซอร์วิสอินสแตนซ์ แท้จริงแล้วยังมีชีวิตอยู่มั้ย และไม่ตอบรับค่าอินสแตนซ์ ว่ามันต้องยกเลิกอย่างไร อย่างไรก็ตาม การมีของ GSR ที่ถูกต้อง ไม่ได้เป็นการแน่นอนว่า กริดเซอร์วิสอินสแตนซ์ นั้นจะติดต่ออย่างไร และ เซอร์วิสบางทีจะล่มหรือ ยกเลิกระหว่างเวลาที่ GSR ไม่ได้ให้ออกมาว่าถูกใช้อยู่ เกี่ยวกับ HandleMap Interface นั้นจะมีปัญหาที่เกี่ยวกับบรรจุ GSR เพื่อให้เป็นเซอร์วิสอยู่ 2 ปัญหา ดังนี้

1) การบอก handleMap service ว่ามันบรรจุการ map จาก GSH ที่เจาะจง

2) การติดต่อกับ handleMap เพื่อให้บรรจุค่า GSR ที่ต้องการเข้าไป

ใน 2 ปัญหานี้ เป็นปัญหาที่เกี่ยวกับการ map GSH เป็น GSR ซึ่งจำเป็นต้องให้ทุกๆกริดเซอร์วิสอินสแตนซ์ จะต้องถูกลงทะเบียนอย่างน้อย 1 ครั้งจาก handleMap ซึ่งเรียกว่า home handleMap โดยโครงสร้างแล้ว GSH จะรวม home handleMap's identity เข้าไปด้วย โดยการติดต่อให้ handleMap บรรจุค่า GSR จากค่า GSH ที่ให้ไป สามารถระบุและกำหนดได้โดยง่ายเนื่องจากว่า มีชื่อที่เป็นเอกเทศในแต่ละ local นั้นๆ เพื่อหลีกเลี่ยงการซ้ำกันของ service ที่ถูกสร้างขึ้นมาจากศูนย์กลาง แม้ว่าจะอาศัย Domain Name System อย่างไรก็ตามทุกๆ GSH นั้นจะต้องมี 1 home handleMap

ในการสร้าง HandleMap นั้น เป็นการใช้กริดเซอร์วิสสร้าง และมันจะต้องมี GSH ให้ จึงใช้ค่า GSH นี้ในการสร้าง ซึ่งจะทำได้ย่อมต้องการ ให้ home handleMap ถูกกำหนดค่าโดย URL และรองรับ bootstrapping operation ที่จะทำให้เป็นหนึ่งเดียว โดยใช้โปรโตคอลที่รู้จักกันดีในการให้ชื่อ เช่น HTTP (หรือ HTTPS) เนื่องจากการใช้ GSR เพื่อบอกว่า โปรโตคอลอะไรที่ควรจะติดต่อกับ handleMap service ซึ่งการใช้ HTTP GET operation ที่ถูกใช้บน URL ที่จุดนั้นเพื่อให้เป็น home handleMap และ GSR สำหรับ handleMap ในรูปแบบ WSDL เพื่อตอบกลับไป

ความสัมพันธ์ระหว่างเซอร์วิส ที่ implement HandleMap และ Factory interface โดยเฉพาะ GSH จะ ตอบกลับโดย factory request ที่ต้องบรรจุ URL ของ home handleMap และ GSH/GSR เพื่อ mapping จะต้องเข้าไปและปรับปรุงค่าได้ใน handleMap serviee ในการ implement faetory จะต้อง

ตัดสินใจว่าจะให้เซอร์วิสอะไรใช้ใน home handleMap ที่จริงแล้วเพียงเซอร์วิสเดียวบางทีจะสามารถสร้างได้ทั้ง Factory และ HandleMap Interface

7.2.5 Service Data และ Service Discovery

ส่วนที่เกี่ยวข้องกับกริดเซอร์วิสอินสแตนซ์นั้นเป็นกลุ่มของข้อมูลที่เรียกว่า service data ซึ่งเป็นกลุ่มของค่า XML ที่ฝังอยู่ในค่าของข้อมูลของเซอร์วิส ซึ่งในแต่ละค่านั้นจะมีชื่อของกริดเซอร์วิสอินสแตนซ์นั้น, ชนิด, เวลาที่ยังมีชีวิตอยู่ ซึ่งเป็นข้อมูลที่ใช้ในกระบวนการจัดการไลฟ์ไทม์

รูปแบบของกริดเซอร์วิสนั้นจะกำหนดด้วยคำสั่ง WSDL โดยใช้ FindServiceData สำหรับการคิวรี่และเลือกเอามาจากข้อมูลของเซอร์วิสนั้น ซึ่งคำสั่งนี้จะใช้ชื่อในการคิวรี่ และอาจขยายไปรูปแบบอื่นๆ เช่น Xquery ที่ใช้กับกรณีพิเศษๆบางกรณี

คุณสมบัติของกริดเซอร์วิสนั้นจะต้องประกอบไปด้วยค่า service data อย่างน้อย 1 ค่าที่รองรับโดยกริดเซอร์วิสอินสแตนซ์ใดๆ ค่าที่ใช้ เช่น GSH,GSR, คีย์ชั้นปฐมภูมิ และค่า handleMap เป็นต้น

ใน 1 แอพพลิเคชั่นของกริดเซอร์วิสนั้นจะต้องมี FindService ที่ใช้ค้นหาเซอร์วิส ซึ่งจะใช้ค่า GSH ในการค้นหา และจะพิจารณาจากค่า การจัดเตรียม, จำนวนที่ร้องขอใช้เซอร์วิส, ค่าโหลดของเซอร์วิส, หรือนโยบายที่ใช้จัดการ มาใช้เลือกเซอร์วิสที่เหมาะสม

กริดเซอร์วิสที่รองรับการค้นหานั้นจะถูกเรียกโดยค่า registry ซึ่งใช้กำหนดอยู่ 2 อย่าง คือรูปแบบการลงทะเบียน ที่จะกำหนดว่า GSH ใดที่จะให้ลงทะเบียนด้วยเซอร์วิสที่ใช้ลงทะเบียนและใช้กำหนดค่าที่เกี่ยวข้องกับการลงทะเบียนของ GSH

การลงทะเบียนนั้นเป็นการยอมให้ GSH ที่ลงทะเบียนไว้ด้วยเซอร์วิสที่ใช้ลงทะเบียนเพื่อคัดเลือกกลุ่มย่อยให้ได้อย่างถูกต้อง ตรงตามความต้องการ เพื่อสะดวกต่อการค้นหาตามที่ร้องขอ ซึ่งค่าที่ลงทะเบียนของ GSH นั้นจะต้องรีเฟรชเป็นระยะๆ เพื่อให้เซอร์วิสที่ใช้ค้นหาสามารถเลือกเซอร์วิสที่เหมาะสมที่สุดได้

7.2.6 Notification

ใน OGSA ส่วนเฟรมเวิร์คของ notification นั้นจะเป็นส่วนที่ยอมให้ไคล์เอนท์จะลงทะเบียนประกาศเฉพาะทางที่สำคัญได้ (ในรูปแบบ NotificationSource) เพื่อจะส่งไปให้ผู้รับตามที่ต้องการ (NotificationSink) ถ้าเซอร์วิสใดๆต้องการให้รองรับข้อความ notification แล้ว มันจะต้องรองรับรูปแบบของ NotificationSource ด้วยซึ่งเซอร์วิสที่ต้องการข้อความ notification จะต้องสร้าง รูปแบบ NotificationSink ขึ้นมาซึ่งใช้ในการส่งข้อความนั้นๆซึ่งขั้นตอนของมันจะเริ่มจากต้องรับรูปแบบของแหล่งข้อมูลก่อน แล้วให้ค่า GSH ของ sink ไปเพื่อใช้ส่งข้อความ โดยที่ sink จะต้องส่งข้อความที่บ่งบอกว่าสนใจข้อความ notification นั้นไปด้วย

สิ่งสำคัญในรูปแบบของ notification นั้นคือต้องปิดการรวมตัวกันด้วยข้อมูลของเซอร์วิส โดยมีคำสั่งพิเศษคือ "push"ที่ใช้ส่งข้อมูลตามเงื่อนพิเศษ (โดยการเรียกคำสั่ง FindServiceData เพื่อเตรียมที่จะ"pull") ซึ่งเฟรมเวิร์คที่ใช้นี้จะยอมให้มีการส่งข้อความตรงๆจากเซอร์วิสไปยังเซอร์วิส รวม

ถึงการส่งไปยังเซอร์วิสกลุ่มที่ 3 ด้วย เช่น การส่งข้อความที่สำคัญทางธุรกิจไปให้กลุ่มที่ต้องการ เป็นต้น โดยต้องมีโปรโตคอลที่รองรับการส่งแบบมัลติคาสท์ด้วย

7.2.7 การควบคุมการเปลี่ยนแปลง

เพื่อรองรับการส่งและการควบคุมการเปลี่ยนแปลงของกริดเซอร์วิส ได้อย่างมีประสิทธิภาพ รูปแบบของกริดเซอร์วิสนั้นจะต้องเป็นแบบที่ทั่วถึงกัน (globally) และมีชื่อเป็นเอกเทศกัน (uniquely) ซึ่งใน WSDL นั้นจะกำหนดไว้ใน portType ด้วย qname โดยการเปลี่ยนแปลงใดๆจะต้องสร้างคำจำกัดความของเซอร์วิสที่เปลี่ยนแปลงนั้น ซึ่งการเปลี่ยนแปลงต้องรองรับกับรูปแบบเดิมด้วย โดยส่วนนี้จะเป็นส่วนที่ยอมให้ไคล์เอนท์ที่ต้องการกริดเซอร์วิสนั้นๆด้วยคุณสมบัติที่พิเศษที่เหมาะสมกับงานของตน เพื่อให้เซอร์วิสนั้นๆใช้ได้อย่างมีประสิทธิภาพ

7.2.8 การผูกกับโปรโตคอลเครือข่าย

ในเฟรมเวิร์คของเว็บเซอร์วิสนั้นจะใช้โปรโตคอลที่แตกต่างกันมาร่วมกันทำงาน เช่น ใช้ SOAP+HTTP พร้อมด้วย TLS สำหรับด้านความปลอดภัย ส่วนนี้จะบ่งบอกว่า OGSA ใช้อะไรบ้าง ซึ่งจะแบ่งเป็น 4 ส่วนที่มีความสำคัญดังนี้

- Reliable transport: อย่างที่กล่าวไปแล้วว่ากริดเซอร์วิสนั้น จำเป็นจะต้องมีการส่งเซอร์วิสไปได้อย่างน่าเชื่อถือ ซึ่งโปรโตคอลที่รองรับด้านนี้ก็เช่น HTTP-R
- Authentication and delegation: จากที่กริดเซอร์วิสนั้นจำเป็นที่จะต้องรองรับการติดต่อสื่อสารไปยังที่อื่นๆ ซึ่งต้องมีการรับรองด้วย ทางหนึ่งที่ใช้กันคือ ใช้การสื่อสารแบบทางเดียว เช่น ใช้ TLS ช่วยรองรับด้านนี้
- Ubiquiry: เมื่อเป้าหมายของกริดคือเพื่อให้เป็นแหล่งรวมของ VO จากรีซอร์สที่กระจายตามส่วนต่างๆ จึงจำเป็นต้องมีการเลือกเซอร์วิสที่เหมาะสมที่จะใช้ตอบโต้กัน
- GSR Format: ในการเรียก GSR นั้น จะต้องมีการผูกกันด้วยรูปแบบพิเศษ 1 ในรูปแบบที่ใช้ก็คือ รูปแบบเอกสาร WSDL แต่ก็มีทางเลือกอื่น เช่น CORBA IOR เป็นต้น

7.2.9 Higher-Level Services

ส่วนทฤษฎีและเซอร์วิสที่กล่าวในหัวข้อนี้ จะบอกถึงการเตรียมสิ่งที่จะใช้สร้างกริดเซอร์วิสระดับสูงขึ้นไป ซึ่งเอาไว้ใช้ในด้าน ธุรกิจ หรือ วิทยาศาสตร์ ซึ่งจะอาศัยส่วนต่างๆ ต่อไปนี้

- Distributed data management service: เป็นส่วนที่รองรับการเข้าถึงและการย้ายของข้อมูลที่กระจายกัน ไม่ว่าจะเป็นฐานข้อมูลหรือแฟ้มข้อมูล ซึ่งเซอร์วิสที่รวมอยู่ในนี้จะมีการเข้าถึงฐานข้อมูล, การแปลงข้อมูล, การจำลองการจัดการ, การจำลองตำแหน่งที่เก็บ และ ทรานแซคชั่น

- Workflow service: เป็นส่วนที่รองรับการทำงานของหลายๆแอพพลิเคชั่นบนรีซอร์สที่กระจายกันอยู่

- Auditing service: เป็นส่วนที่รองรับการจำข้อมูล ทำให้ที่เก็บข้อมูลมีความปลอดภัย วิเคราห์ถึงข้อมูลที่มีจุดประสงค์ไม่ดีหรือบุกรุกเข้ามา

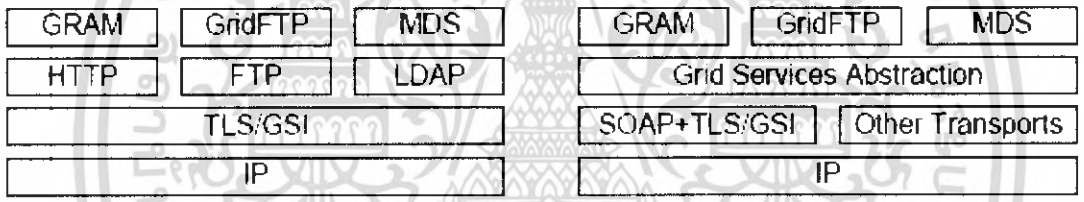
- Instrumentation and monitoring service: รองรับการค้นหาในสภาพแวดล้อมแบบกระจาย โดยจะอาศัย เซ็นเซอร์ช่วย ซึ่งจะมีหน้าที่รวบรวมข่าวสารและวิเคราะห์ เพื่อใช้เตือนสิ่งที่ผิดปกติด้านต่างๆ

- Problem determination service for distributed computing: รวมหลายๆวิธีการ เช่น dump,trace และ log ด้วยการบันทึกเหตุการณ์และด้วยความสามารถในการเข้าคู่กัน

- Security protocol mapping services: จัดเตรียมโปรโตคอลที่ปลอดภัยสำหรับการกระจายส่วนต่างๆ โดยจะ map เข้ากับรูปแบบทั่วไป โดยมีการรับรองความปลอดภัยในการกระจายและการควบคุมสิทธิการเข้าถึงส่วนต่างๆ

ความยืดหยุ่นของเฟรมเวิร์คนี้ก็คือ สามารถสร้างและรวมได้จากหลากหลายวิธี เช่น รวมเซอร์วิสจากส่วนที่จองไว้และใช้หลายๆรีซอร์สมาช่วยในการคำนวณ หรือ อาจจะเชื่อมกับแอพพลิเคชั่นที่เป็นไลบรารี่ หรือ รวมกับเซอร์วิสระดับสูง เป็นต้น

ดังนั้น เพื่อความสะดวกในการจัดการรีซอร์ส, การขนส่งข้อมูล, การใช้โปรโตคอลของเซอร์วิสใน Globus Toolkit นั้นจึงสร้างเครื่องมือในด้านต่างๆไว้ ดังภาพด้านล่างนี้



รูปที่ 7-2 ภาพทางซ้ายเป็นโปรโตคอลของ Globus Toolkit 3 ส่วนภาพทางขวาเป็นแนวคิดของ OGSA

7.3 Open Grid Service Infrastructure

OGSI (Open Grid Services Infrastructures) นั้นเป็นส่วนที่คิดขึ้นมาเพื่อใช้กำหนดคุณสมบัติของระบบกริด ซึ่งจะกล่าวถึง ความสัมพันธ์ระหว่างโครงสร้างของกริด, ระบบการคำนวณแบบกระจาย, ความเกี่ยวพันกับเว็บเซอร์วิส เฟรมเวิร์ค และความเกี่ยวพันระหว่าง client กับ hosting environment

7.3.1 ความสัมพันธ์กับระบบการคำนวณแบบกระจาย

กริดเซอร์วิส (Grid Service) นั้นจะสร้างขึ้นมาแบบฝังไว้ภายในแต่ละอินสแตนซ์ (instance) ซึ่งมีส่วนที่ใช้อธิบายอินสแตนซ์ ก็คือ WSDL portType โดยหลายๆ Grid Service นั้นจะรวมตัวกัน เรียกว่า Grid Services Factory ซึ่งใช้สร้างอินสแตนซ์ สำหรับแต่ละ portType โดยที่แต่ละอินสแตนซ์ จะเป็นส่วนที่ใช้ติดต่อกันระหว่าง 2 กริดเซอร์วิส ในการทำงานแต่ละครั้งก็จะติดต่อไปที่อินสแตนซ์ ให้ส่งค่าที่ต้องการกลับมา

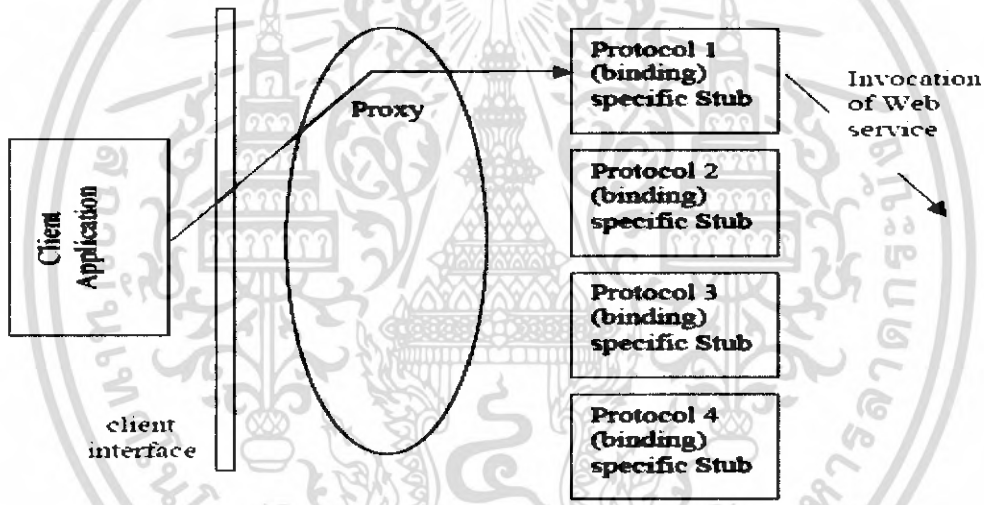
กริดเซอร์วิสอินสแตนซ์นั้น ใช้เพื่อให้เข้าถึงแอพพลิเคชั่นของไคล์เอนท์โดยอาศัย Grid Service Handle และ Grid Service Reference เพื่อส่งรีเควสต์ตรงไปที่แต่ละอินสแตนซ์ที่เป็นเป้าหมายซึ่งจะได้

รับคำตอบเป็น เครื่องเป้าหมายที่จะส่งงานไปให้คำนวณ แสดงออกมาเป็นค่า Grid Service Reference ที่ใช้ในการติดต่อกันจริง

โดยสรุปแล้วสิ่งที่เรียกว่าระบบการคำนวณแบบกระจายในกริดนั้น จะอาศัย GSR ในการกำหนดว่าควรจะส่งงานไปให้ที่ส่วนไหนทำงานโดยอาศัยตัวจัดการที่เรียกว่าตัวจัดเตรียมเซอร์วิส (Service Provider) ที่เก็บรวบรวมข้อมูลว่า แต่ละโฮสต์ในกริด นั้นมีเซอร์วิสที่มีไคล์เอนท์ต้องการเรียกใช้หรือไม่ ถ้าเครื่องไหนมีและพบว่ามีความพร้อมที่สุด ก็จะส่งไปเครื่องนั้นๆทำงาน

7.3.2 แนวทางการเขียนโปรแกรมฝั่งไคล์เอนท์

เนื่องจากกริดในยุคหลังนั้น ได้นำเทคโนโลยีของเว็บเซอร์วิส มาช่วยในการพัฒนาส่วนต่างๆ โดยส่วนที่ OGSI นำเว็บเซอร์วิสใช้นั้น ส่วนหนึ่งก็คือ WSDL ที่ใช้อธิบายว่า ผูกกับโปรโตคอลอย่างไร, เอนโค้ดอยู่ในรูปแบบใด, ส่งข้อความแบบใด (ระหว่าง RPC หรือ document-oriented) และใช้เว็บเซอร์วิส อย่างไร ซึ่งในกริด จะใช้ Web Service Invocation Framework ร่วมกับ Java API for XML RPC ในหลายๆส่วนของซอฟท์แวร์ ที่พัฒนาขึ้น

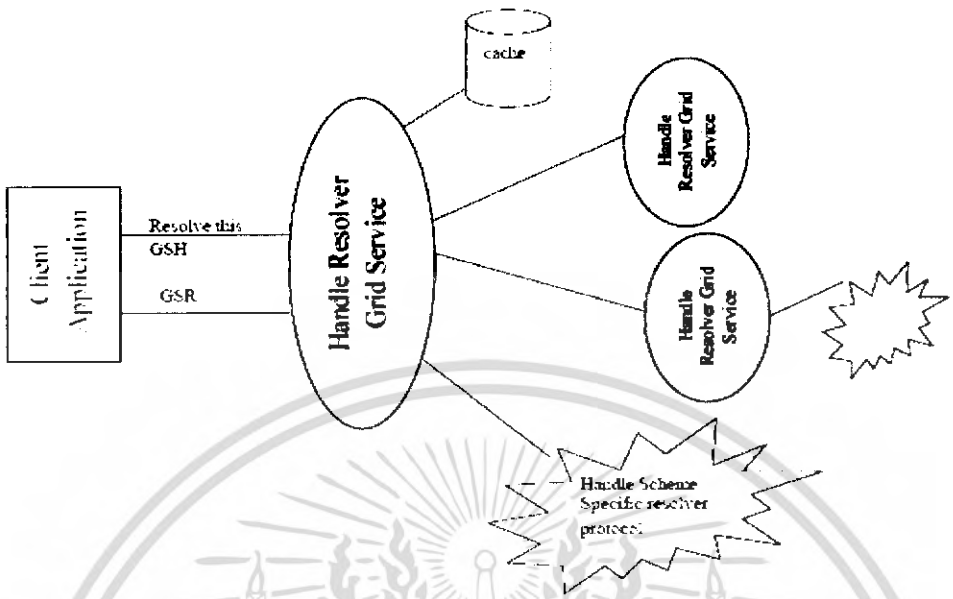


รูปที่ 7-3 แสดงโครงสร้างการทำงานฝั่งไคล์เอนท์

7.3.3 การใช้ Grid Services Handles และ References ของไคล์เอนท์

การที่ไคล์เอนท์จะเข้าถึงกริดเซอร์วิสอินสแตนซ์ได้นั้น จะต้องผ่าน GSH (Grid Service Handle) และ GSR (Grid Service Reference) โดยที่ GSH นั้นจะเปรียบเสมือน Network Pointer ที่จะชี้ไปที่กริดเซอร์วิสอินสแตนซ์ แต่ GSH ไม่ใช่ส่วนที่จัดเตรียมข้อมูลที่จะให้ไคล์เอนท์เข้าถึงเซอร์วิสอินสแตนซ์ได้การที่จะเข้าถึง กริดเซอร์วิสอินสแตนซ์ได้นั้นไคล์เอนท์จะต้องผ่านวิธีการแปลง GSH ให้เป็น GSR (Grid Service Reference) ซึ่ง GSR จะเป็นส่วนเก็บข้อมูลที่จำเป็นสำหรับการเข้าถึงกริดเซอร์วิสอินสแตนซ์ ไว้ โดยที่ GSR ไม่ใช่ Network Pointer ที่ชี้ไปที่กริดเซอร์วิสอินสแตนซ์ เนื่องจาก GSR อาจทำให้ผิดพลาดได้ในบางกรณี เช่น กริดเซอร์วิสอินสแตนซ์ นั้นย้ายไปที่ Server ตัวอื่นแล้ว เป็นต้น

OGSI นั้นได้จัดเตรียมวิธีการที่จะทำ HandleResolver เพื่อรองรับให้ไคล์เอนท์สามารถแปลง GSH ให้เป็น GSR ได้ ในรูปด้านล่างจะแสดงวิธีการให้ดู

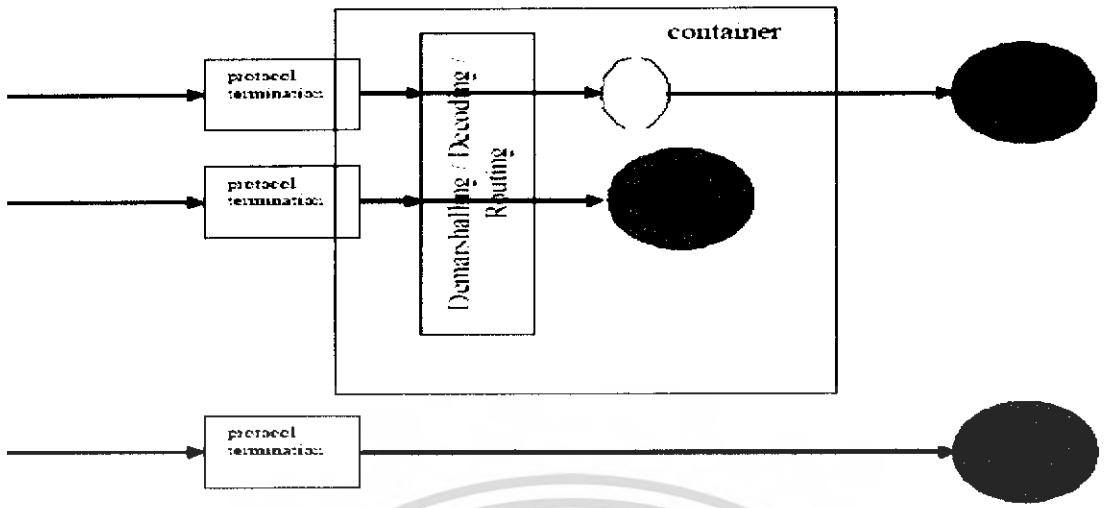


รูปที่ 7-4 แสดงวิธีการ resolve GSH

ไคล์เอนท์นั้นจะแปลงจาก GSH เป็น GSR โดยผ่านการ invoke ที่ HandleResolver กริดเซอร์วิสอินสแตนซ์โดยที่ HandleResolver จะมีค่า GSR เก็บไว้ที่แคช ภายในของตัวมัน แต่ถ้าไม่มีเก็บไว้ที่แคชมันก็ต้องขอ HandleResolver ตัวอื่นๆให้ช่วยติดต่อไปที่ GSH ที่ต้องการให้ โดยอาจจะผ่านโปรโตคอลที่นิยมใช้กันทั่วๆไป เช่น ใช้ HTTP เพื่อรับค่า URL แล้วจึงค่อยเอนโค้ดจากค่า GSH ให้เป็น GSR เป็นต้น

7.3.4 ความสัมพันธ์ระหว่างไคล์เอนท์กับ Hosting Environment

OGSI นั้นไม่ได้ทำส่วนในฝั่งที่เป็นตัวจัดเตรียมเซอร์วิส (Service Provider) แต่จะใช้โมเดลคล้ายๆกับของฝั่ง Server ที่ใช้ใน J2EE กล่าวคือ ทุกๆส่วนในมาตรฐานของกริดเซอร์วิส (เช่น ส่วนการร้องขอ, ส่วนจัดการไลฟ์ไทม์, ส่วนที่เกี่ยวกับการลงทะเบียน เป็นต้น) จะถูกซ่อนไว้ภายใน ยูสเซอร์โปรเซส เช่น การเชื่อมต่อกับไลบรารี่มาตรฐานเป็นต้น



รูปที่ 7-5 แสดงลักษณะการทำงานของฝั่ง hosting environment

จากรูป กล่าวคือ กริดเซอร์วิสอินสแตนซ์นั้น จะฝังอยู่ในคอนเทนเนอร์ ที่ข้อความต่างๆ จะผ่านภายใน คอมโพเนนท์เหล่านี้ ในรูปแบบต่างๆ เช่น XML หรือ SOAP เป็นต้น

โดยส่วนมากแล้ว ในคอนเทนเนอร์นั้นมักจะสร้างขึ้นมาให้ใช้ในเรื่องต่างๆ เช่น จัดเตรียมการจัดการ เวลาชีวิต, หรือการตรวจสอบสิทธิและอนุญาตแบบอัตโนมัติ, การร้องขอเพื่อเข้าสู่ระบบ, การยกเลิกเซอร์วิสอินสแตนซ์เมื่อหมดไลฟ์ไทม์ และ ยกเลิกการร้องขอแบบตรงๆ

7.3.5 คุณสมบัติของกริดเซอร์วิส

จากที่กล่าวไปบทก่อนๆแล้ว ว่าทุกอย่างที่เป็นกริดเซอร์วิสก็คือเว็บเซอร์วิสแม้ว่าลักษณะมันจะไม่เหมือนกันทุกอย่างก็ตาม ในส่วนนี้จะระบุคุณสมบัติของกริดเซอร์วิสไว้

- ใช้ส่วนของ WSDL ตามที่เว็บเซอร์วิสใช้ ตั้งแต่ WSDL 1.2 ขึ้นไป
- กำหนด เซอร์วิสเดต้า (service data) ซึ่งเป็นตัวมาตรฐานที่มาจากแสดงหรือซักถามจากเซอร์วิสอินสแตนซ์
- แนะนำเกี่ยวกับแกนในคุณสมบัติของกริดเซอร์วิส ที่ต้องมี
  - กำหนดกริดเซอร์วิสเคสคริปชั่น(Grid service description) และกริดเซอร์วิสอินสแตนซ์จากลักษณะการใช้งานจริงๆ
  - กำหนดว่า OGSI มีรูปแบบเกี่ยวด้านเวลาอย่างไร
  - กำหนดการสร้าง GSH และ GSR ซึ่งใช้ในการอ้างกริดเซอร์วิสอินสแดนซ์
  - กำหนดวัฏจักรชีวิตของกริดเซอร์วิสอินสแตนซ์

7.4 เซอร์วิสเดต้า (Service Data)

การที่กริดเข้าไปใกล้การเป็นเว็บเซอร์วิสที่ทำงานเป็นสเตท ทำให้มันจำเป็นต้องมีกลไกที่จะใช้อธิบายข้อมูลในสเตทของเซอร์วิสอินสแตนซ์ที่ผู้ร้องขอคิวรี่, อัพเดต, เปลี่ยนการประกาศ อย่างที่เว็บเซอร์วิสทำ ตั้งแต่ที่ใช้เว็บเซอร์วิสรวมสิ่งที่ใช้ภายนอกซึ่งใช้บ่งบอกข้อมูลสเตทที่เรียกว่า เซอร์วิสเดต้า

ในการเตรียมส่วนที่ใช้อธิบายรายละเอียดของเว็บเซอร์วิสที่ทำงานเป็นเสตท (เช่น กริดเซอร์วิส) มันจำเป็นที่จะต้องอธิบายสเตทที่ใช้กับส่วนภายนอกอย่างชัดเจนซึ่งก็คือสเตทที่เซอร์วิสอินสแตนซ์ถูกแสดงเมื่อผู้ใช้สร้างเซอร์วิสขึ้นมานั่นเอง สิ่งที่จำเป็นสำหรับการประกาศเซอร์วิสเดต้าก็คือแนวความคิดที่จะประกาศคุณสมบัติในตัวมัน ซึ่งอธิบายได้จาก Interface Definition Language (IDL) โดยที่เซอร์วิสเดต้านั้นจะสามารถใช้ได้จากการอ่าน, อัพเดต หรือ ลงท้ายในข้อมูล

ซึ่ง WSDL นั้นได้กำหนด กระบวนการและข้อความสำหรับ portTypes ที่สเตทที่ถูกประกาศนั้นต้องมีการเข้าถึงได้จากภายนอกผ่านกระบวนการของเซอร์วิสที่ใช้กำหนดเซอร์วิสอินเตอร์เฟสเท่านั้น เพื่อหลีกเลี่ยงความต้องการที่จะกำหนด กระบวนการพิเศษของเซอร์วิสเดต้าสำหรับแต่ละเซอร์วิสเดต้า โดยที่กริดเซอร์วิสนั้น จะมี portType ที่ใช้จัดกระบวนการที่ใช้ทำเซอร์วิสเดต้าด้วยตัวเอง

ตัวอย่าง มีอินเตอร์เฟสที่มีกระบวนการ op1, op2, และ op3 อยู่ โดยสมมติให้อินเตอร์เฟสนี้มีข้อมูลที่ให้สาธารณะเข้าถึงได้คือ de1, de2, de3 ซึ่งผู้ใช้ได้กำหนดรายละเอียดของอินเตอร์เฟสนี้ด้วย WSDL ซึ่ง serviceData ใน OGSI นี้ จะสร้างสืบเนื่องมาจาก WSDL ซึ่งผู้ออกแบบจะสามารถเพิ่มรายละเอียดอินเตอร์เฟสนี้ได้โดยกำหนดส่วนที่เกี่ยวกับการยอมให้เข้าถึงแบบสาธารณะของแต่ละสเตทสำหรับ de1, de2, de3 โดยการประกาศนี้จะทำให้สะดวกต่อการใช้งานของกระบวนการต่างๆบนเซอร์วิสเดต้าของเซอร์วิสอินสแตนซ์แบบมีสเตทซึ่งใช้พัฒนาอินเตอร์เฟสนี้ขึ้นมา

7.4.1 การเปรียบเทียบกับ Java Bean

ในคุณสมบัติของ serviceData ที่ OGSI ได้กำหนดสเป็คให้เป็นส่วนที่เข้าถึงข้อมูลสเตทของเว็บเซอร์วิส โดยแนวคิดของ serviceData นั้น คล้ายคลึงกับของ public instance variable หรือ field ใน object-oriented programming language เช่น Java, Smalltalk หรือ C++

ServiceData นั้นคล้ายคลึงกับคุณสมบัติของ JavaBean ซึ่งโมเดลของ JavaBean นั้นได้กำหนดรูปแบบสำหรับการทำ method signatures (getXXX/setXXX) เพื่อเข้าถึงคุณสมบัติ และผู้ช่วยในการสร้างคลาส (BeanInfo) เพื่อทำคุณสมบัติของเอกสาร โดย OGSI ได้ใช้ส่วนของ serviceData และ XML สำหรับการทำในสิ่งที่คล้ายๆกันกับของ JavaBean

โดยในสเป็คของ OGSI นั้นไม่ได้เลือกใช้วิธี getXXX และ setXXX เพื่อใช้เป็นกระบวนการของ WSDL ที่นำมาใช้สำหรับแต่ละ serviceData แม้ว่าส่วนที่ใช้พัฒนาเซอร์วิสจะเลือกวิธี get และ set ด้วยตัวมันเองก็ตาม โดย OGSI ได้เลือกใช้กระบวนการ query แทนการ get และใช้วิธีการ update แทนการ set และได้ใช้ subscribe to notification สำหรับแก้ไขค่าใน serviceData ซึ่งสิ่งที่ OGSI ต้องการนั้นอย่างน้อยต้องรองรับวิธีการเหล่านี้ ซึ่งเป็นสิ่งที่อนุญาตให้เข้าถึง serviceData ในชื่อของมันของแต่ละเซอร์วิสอินสแตนซ์ อย่างไรก็ตามกระบวนการบางส่วนของ OGSI ก็นำบางส่วนมาจากอินเตอร์เฟส

อื่นๆ เพื่อรองรับการคิวรี่, อัพเดต และ ลงท้าย แบบที่ยุ่งยากขึ้น เช่นการ คิวรี่หลายๆ serviceData ใน 1 เซอร์วิสอินสแตนซ์

โดยค่า serviceDataName ใน GridService portType นั้น ได้กำหนดไว้ตามรูปแบบเดียวกับของ BeanInfo ใน JavaBeans อย่างไรก็ตาม OGSI ได้เลือก XML (WSDL) สำหรับจัดเตรียมข้อมูลข่าวสารเกี่ยวกับ serviceData แทนที่การใช้โมเดลของ BeanInfo แบบตรงๆ

7.4.2 การแทนค่า portType ด้วย serviceData

ServiceData ที่ใช้กำหนดค่าของ portType นั้นจะเรียกว่า ServiceData Elements (SDEs) ซึ่งใน SDE จะใช้กำหนดการอ้างจาก ServiceData Declarations (SDDs) โดยค่าเริ่มต้นของ SDE จะกำหนดได้จากค่า staticServiceDataValues ภายใน portType โดยค่าใดๆของ SDE จะถูกกำหนดใน portType หรืออาจจะถูกใส่ค่าจากเว็บเซอร์วิสอินสแตนซ์

ส่วนนี้เป็นตัวอย่างการใช้ gwsdl:portType ซึ่งยอมให้มีค่าปรากฏจาก namespaces อื่นๆ

```
<gwsdl:portType name="NCName"> *
   <wsdl:documentation … /> ?
   <wsdl:operation name="NCName"> … </wsdl:operation> ?
…
   <sd:serviceData name="NCName" … /> *
   <sd:staticServiceDataValues>?
      <some element>*
   </sd:staticServiceDataValues>
…
</gwsdl:portType>
```

สำหรับตัวอย่างนี้ จะเป็นการประกาศ 2 SDEs ของ portType ด้วยชื่อ “tns:sdl” and “tns:sdl2” ซึ่งไม่ว่าเซอร์วิสอินสแตนซ์ใดๆที่สร้างจาก portType นี้จะต้องมีค่า 2 ค่านี้

```
<wsdl:definitions xmlns:tns="xxx" targetNamespace="xxx">
   <gwsdl:portType name="exampleSDUse"> *
      <wsdl:operation name=…> … </wsdl:operation>
…
      <sd:serviceData name="sd1" type="xsd:String"
                      mutability="static"/>
      <sd:serviceData name="sd2" type="tns:SomeComplexType"/>
…
      <sd:staticServiceDataValues>
         <tns:sdl>initValue</tns:sdl>
      </sd:staticServiceDataValues>
   </gwsdl:portType>
…
</wsdl:definitions>
```

7.4.2.1 โครงสำหรับการประกาศ serviceData

จากที่ sd:serviceData ใช้กำหนดค่าใน XMML ซึ่งปรากฏใน sd:serviceDataValue และ sd:serviceData ที่ได้ออกแบบไว้หลังจากกำหนด xsd:element ของ XML ซึ่งค่าใน sd:serviceData จะใช้ 5 แอททริบิวท์เหมือนกันอย่างที่แสดงใน xsd:element: ประกอบด้วย name, type, minOccurs, maxOccurs, และ nillable ซึ่ง xsd:element อื่นๆจะห้ามใช้ภายใน sd:serviceData ซึ่งค่า sd:serviceData จะอนุญาตให้ 2 ค่าพิเศษนี้เพิ่มเติมเข้าไป ก็คือค่า mutability และค่า modifiable

โครง XML ที่ใช้กำหนด sd:serviceData นั้น ดูได้ตามนี้

```
-
    targetNamespace =
      "http://www.gridforum.org/namespaces/2003/serviceData"
    xmlns:sd =
      "http://www.gridforum.org/namespaces/2003/03/serviceData"
-

<attributeGroup name="occurs">
   <attribute name="minOccurs"
              type="nonNegativeInteger"
              use="optional"
              default="1"/>
   <attribute name="maxOccurs">
      <simpleType>
         <union memberTypes="nonNegativeInteger">
            <simpleType>
               <restriction base="NMTOKEN">
                  <enumeration value="unbounded"/>
               </restriction>
            </simpleType>
         </union>
      </simpleType>
   </attribute>
</attributeGroup>

<complexType name="ServiceDataType">
   <sequence>
      <any namespace="##any" minOccurs="0" maxOccurs="unbounded"/>
   </sequence>
   <attribute name="name" type="NCName"/>
   <attribute name="type" type="QName"/>
   <attribute name="nillable"
              type="boolean"
              use="optional"
              default="false"/>
   <attributeGroup ref="sd:occurs"/>
   <attribute name="mutability" use="optional" default="extendable">
      <simpleType>
         <restriction base="string">
            <enumeration value="static"/>
            <enumeration value="constant"/>
            <enumeration value="extendable"/>
            <enumeration value="mutable"/>
         </restriction>
      </simpleType>
   </attribute>
   <attribute name="modifiable" type="boolean" default="false"/>
   <anyAttribute namespace="##other" processContents="lax"/>
</complexType>
```

```
<element name="serviceData" type="sd:ServiceDataType"/>

<xsd:complexType name="ServiceDataValuesType">
   <xsd:sequence>
      <xsd:any namespace="##any" minOccurs="0"
                                 maxOccurs="unbounded" />
   </xsd:sequence>
</xsd:complexType>

<element name="serviceDataValues"
         type="sd:ServiceDataValuesType"/>

<element name="staticServiceDataValues"
         type="sd:ServiceDataValuesType"/>
```

โดยค่าของ แอททริบิวท์แต่ละส่วน serviceData คือตามนี้

- maxOccurs = (nonNegativeInteger | unbounded) : default to 1

  - ค่านี้แสดงค่าตัวเลขที่มากที่สุดของ serviceData ที่สามารถปรากฏได้ในค่า serviceDataValues ของเซอร์วิสอินสแตนซ์ หรือ ค่า staticServiceDataValues ของ portType

- minOccurs = nonNegativeInteger : default to 1

  - ค่านี้แสดงค่าตัวเลขที่น้อยที่สุดของ serviceData ที่สามารถปรากฏได้ในค่า serviceDataValues ของเซอร์วิสอินสแตนซ์ หรือ ค่า statieServiceDataValues ของ portType

  - ถ้าค่านี้เป็น 0 แสดงว่าค่าของ serviceData นี้เป็นแบบ optional

- name = NCName and {target namespace}

  - ชื่อของ serviceData ต้องเป็นเอกเทศกันระหว่าง sd:serviceData และ xsd:element ทั้งหมดที่ใช้ประกาศใน namespace เป้าหมายของ wsdl:definition element

  - การรวมกันของชื่อใน serviceData และ targetNamespace ของ wsdl:element จาก QName นั้นจะยอมให้มีการอ้างเอกเทศของค่า serviceData นี้

- nillable = boolean : default to false

  - ค่านี้จะแสดงว่า serviceData มีค่าที่เป็น nil (ซึ่งเป็นค่าที่มี xsi:nil เป็นค่า true)

  ตัวอย่าง

  <serviceDataElement name="foo" type="xsd:string"

  nillable=true" />

  <foo xsi:nil="true"/>

- type = QName

  - ค่านี้ใช้กำหนดชนิดของ XML ของ serviceData

- modifiable = “boolean” : default to false

  - ถ้าค่านี้เป็นจริง จะหมายความว่าเป็นผู้ร้องขอโดยตรงเพื่อขออัพเดตค่า serviceData ผ่านคำสั่ง serServiceData ซึ่งเป็นการสั่งแบบฝืนค่า cardinality (minOccurs, maxOccurs) และ mutability ถ้าค่านี้เป็นเท็จ จะหมายถึงค่าของ serviceData ควรจะถูกพิจารณาแบบ read only เท่านั้นโดยผู้ร้องขอ แม้ว่าค่านั้นจะเปลี่ยนผลลัพธ์ของกระบวนการอื่นๆบนอินเตอร์เฟสนี้ก็ตาม

- mutability = “static” | “constant” |“extendable” | “mutable” : default to extendable

  - ค่านี้แสดงว่า serviceData จะสามารถเปลี่ยนได้อย่างไร

7.4.2.2 การใช้ serviceData โดยใช้ตัวอย่างจาก GridService portType

เพื่อแสดงว่า serviceData ถูกใช้อย่างไร จะอธิบายได้โดยโค้ดส่วนนี้

```
<wsdl:definitions …

<gwsdl:portType name="GridService" …>
   <wsdl:operation name=…> … </wsdl:operation>
   …

   <sd:serviceData name="interface" type="xsd:QName"
        minOccurs="1" maxOccurs="unbounded"
        mutability="constant"/>
   <sd:serviceData name="serviceDataName" type="xsd:QName"
        minOccurs="0" maxOccurs="unbounded"
        mutability="mutable" nillable="false"/>
   <sd:serviceData name="factoryHandle"
        type="ogsi:HandleType"
        minOccurs="1" maxOccurs="1"
        mutability="constant" nillable="true"/>
   <sd:serviceData name="gridServiceHandle"
        type="ogsi:HandleType"
        minOccurs="0" maxOccurs="unbounded"
        mutability="extendable"/>
   <sd:serviceData name="gridServiceReference"
        type="ogsi:ReferenceType"
        minOccurs="0" maxOccurs="unbounded"
        mutability="mutable"/>
   <sd:serviceData name="findServiceDataExtensibility"
        type="ogsi:OperationExtensibilityType"
        minOccurs="1" maxOccurs="unbounded"
        mutability="static"/>
   <sd:serviceData name="terminationTime" type="ogsi:terminationTime"
        minOccurs="1" maxOccurs="1"
        mutability="mutable"/>
   <sd:serviceData name="setServiceDataExtensibility"
        type="ogsi:OperationExtensibilityType"
        minOccurs="2" maxOccurs="unbounded"
        mutability="static" />
…
```

ส่วนนี้คือตัวอย่างการเซ็ตค่า serviceData สำหรับกริดเซอร์วิสอินสแตนซ์

```
-
    xmlns:crm="http://gridforum.org/namespaces/2002/11/crm"
    xmlns:tns="http://example.com/exampleNS"
    xnlns="http://example.com/exampleNS">

<sd:serviceDataValues>
    <ogsi:interface>crm:GenericOSPT</ogsi:interface>
    <ogsi:interface>ogsi:GridService</ogsi:interface>

    <ogsi:serviceDataName>ogsi:interface
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:serviceDataName
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:factoryHandle
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:gridServiceHandle
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:gridServiceReference
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:findServiceDataExtensibility
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:terminationTime
    </ogsi:serviceDataName>
    <ogsi:serviceDataName>ogsi:setServiceDataExtensibility
    </ogsi:serviceDataName>

    <ogsi:factoryHandle>someURI</ogsi:factoryHandle>

    <ogsi:gridServiceHandle>someURI</ogsi:gridServiceHandle>
    <ogsi:gridServiceHandle>someOtherURI</ogsi:gridServiceHandle>

    <ogsi:gridServiceReference>…</ogsi:gridServiceReference>
    <ogsi:gridServiceReference>…</ogsi:gridServiceReference>

    <ogsi:findServiceDataExtensibility
          inputElement="ogsi:queryByServiceDataNames" />

    <ogsi:terminationTime after="2002-11-01T11:22:33"
                          before="2002-12-09T11:22:33"/>

    <ogsi:setServiceDataExtensibility
          inputElement="ogsi:setByServiceDataNames" />
    <ogsi:setServiceDataExtensibility
          inputElement="ogsi:deleteByServiceDataNames" />

</sd:serviceDataValues>
```

7.4.3 Mutability

ค่า mutability บน serviceData นั้นจะเป็นส่วนที่แสดงว่าค่าของ serviceData นั้นจะเปลี่ยนได้อย่างไรในไลฟ์ไทม์

mutability = “static” แสดงว่า ค่า SDE ถูกแทนให้ด้วยการประกาศของ WSDL (staticServiceDataValue) และต้องคงมีค่า instance อื่นๆของ portType นั้น โดยค่า “static” นั้นเปรียบเสมือนค่าตัวแปรที่เป็นสมาชิกของคลาสในภาษาโปรแกรมมิ่ง

mutability = "constant" แสดงว่า SDE ถูกแทนให้โดยการสร้างกริดเซอร์วิสอินสแตนซ์และต้องไม่มีการเปลี่ยนแปลงระหว่างช่วงที่อยู่ในเวลาชีวิตของกริดเซอร์วิสอินสแตนซ์นั้นจนกระทั่งมีการเซ็ตค่า

mutability = "extendable" แสดงว่าอย่างน้อย 1 ค่าของ SDE นั้นเป็นส่วนหนึ่งของไลฟ์ไทม์ของ กริดเซอร์วิสอินสแตนซ์ ซึ่งมีค่าใหม่เพิ่มขึ้นมาได้ แต่ค่าเหล่านั้นต้องไม่ถูกลบออกไป

mutability = "mutable" แสดงว่าค่าใดๆใน SDE อาจถูกลบไปบางช่วงเวลาและบางทีอาจถูกเพิ่มขึ้นมา

7.4.4 serviceDataValues

แต่ละเซอร์วิสอินสแตนซ์จะเกี่ยวข้องกับกลุ่มของ serviceData ซึ่งค่าใน serviceData จะถูกกำหนดโดยหลายๆ portType จากอินเตอร์เฟสของเซอร์วิสนั้น และ บางทีอาจจะมีการเพิ่มค่าระหว่างช่วงเวลาที่รันอยู่ ซึ่งเรียกกลุ่มของ serviceData ที่เกี่ยวข้องกับเซอรีวิสอินสแตนซ์นี้ว่า "serviceData set" ซึ่งบางที จะอ้างอิงกับกลุ่มของ serviceData ที่ ไปรวมกับค่าของ serviceData ที่ถูกประกาศไว้ด้วยอินเตอร์เฟสของ portType

แต่ละเซอร์วิสอินสแตนซ์นั้นต้องมีเอกสาร XML ที่เป็นแบบลอจิค พร้อมด้วยค่า serviceDataValues ที่บรรจุ serviceData ไว้ ซึ่งการเขียนเซอร์วิสนั้นจะอิสระที่จะเลือกใช้ว่าค่า SDE นั้นถูกเก็บไว้แบบไหน เช่น บางทีอาจจะไม่ได้เก็บไว้ในรูป XML แต่เก็บเป็นตัวแปรอินสแตนซ์ซึ่งอาจเปลี่ยนรูปจาก XML หรือ วิธีการเอนโค้ดแบบอื่นๆเท่าที่จำเป็น

7.4.5 การกำหนดค่าเริ่มต้นของ SDE

ในการใช้ค่าของ staticServiceDataValues นั้น portType จะต้องมีการประกาศค่าเริ่มต้นสำหรับ serviceData ใดๆก็ตามด้วย mutability ที่เซ็ตไว้แบบ "static" เท่านั้น แม้ว่า servieeData จะถูกกำหนดไว้แบบ local หรือต่อเนื่องมาจาก portTypes ค่าเริ่มต้นนั้นจะต้องกำหนดในหลายๆ portTypes ข้างในอินเตอร์เฟสนั้นๆ ตราบที่ค่าเริ่มต้นนั้นยังไม่เกินค่า maxOccurs ที่ระบุไว้ ตัวอย่างนี้จะเป็นการกำหนดค่าเริ่มต้น 2 ค่าสำหรับ tns:otherSD ให้เป็น "1" และ "2"

```
<wsdl:definitions xmlns:tns="xxx" targetNamespace="xxx">
   <gwsdl:portType name="otherPT">
      <wsdl:operation name=…> … </wsdl:operation>
…
      <sd:serviceData name="otherSD" type="xsd:String"
                      mutability="static" maxOccurs="unbounded"/>

      <sd:staticServiceDataValues>
         <tns:otherSD>initial value 1</tns:otherSD>
      </sd:staticServiceDataValues>
…
   </gwsdl:portType>

   <gwsdl:portType name="exampleSDUse" extends="tns:otherPT">
      <wsdl:operation name=…> … </wsdl:operation>
…
      <sd:serviceData name="sd1" type="xsd:String"
                      mutability="static" />
      <sd:serviceData name="sd2" type="tns:SomeComplexType"/>

      <sd:staticServiceDataValues>
         <tns:sd1>an initial value</tns:sd1>
         <tns:otherSD>initial value 2</tns:otherSD>
      </sd:staticServiceDataValues>
…
   </gwsdl:portType>
…
</wsdl:definitions>
```

7.4.6 การรวมกันของ SDE ภายในโครงสร้างของ portType

WSDL 1.2 นั้นจะแนะนำเกี่ยวกับหลายๆ portType และการออกแบบภายในขอบเขตของ gwsdl ซึ่ง portType จะสามารถสืบมาจาก 0 หรือมากกว่านี้จาก portTypes อื่นๆ โดยที่ไม่ใช่ความสัมพันธ์ตรงๆระหว่าง wsdl:Service และ portType ที่ถูกรับรองโดยโมเดลของเซอร์วิสในรูปแบบของ WSDL ซึ่งกลุ่มของ portType นั้นจะสร้างโดยเซอร์วิสที่สืบทอดมาจากค่าลูกของเซอร์วิสและค่าที่อ้างจากค่า port โดยค่า กลุ่มของ portTypes และ portTypes จะถูกกำหนดด้วยเซอร์วิสอินเตอร์เฟสที่สมบูรณ์

serviceData ที่ถูกกำหนดโดยเซอร์วิสนั้นเป็นเซ็ตที่รวมกันของ serviceData ที่รวมกันของแต่ละ portType ในอินเตอร์เฟสที่สมบูรณ์ที่สร้างมาจากเซอร์วิสอินสแตนซ์ เพราะว่าค่า serviceData จะเป็นเอกเทศจาก QName ซึ่งเป็นกลุ่มจะปรากฏขึ้นเพียงครั้งเดียวในเซ็ตของ serviceData ยกตัวอย่างเช่น ถ้า portType มีชื่อ “pt1” และ “pt2” ซึ่งทั้งคู่กำหนดโดย serviceData ที่ชื่อ “tns:sd1” และมี portType ที่ชื่อ “pt3” ซึ่งสืบทอดมาจาก “pt1” และ “pt2” โดยที่มีแค่ 1 serviceData เท่านั้นที่ชื่อ “tns:sd1”

พิจารณาได้จากตัวอย่างนี้

```
<gwsdl:portType name="pt1">
   <sd:serviceData name="sd1" … />
</gwsdl:portType>

<gwsdl:portType name="pt2" extends="pt1">
   <sd:serviceData name="sd2" … />
</gwsdl:portType>

<gwsdl:portType name="pt3" extends="pt1">
   <sd:serviceData name="sd3" … />
</gwsdl:portType>

<gwsdl:portType name="pt4" extends="pt2 pt3">
   <sd:serviceData name="sd4" … />
</gwsdl:portType>
```

ซึ่งกลุ่มของ serviceData ได้กำหนด 4 portTypes ดังที่แสดงไว้ดังตารางข้างล่าง

if a service implements…	its serviceData set contains…
Pt1	sd1
Pt2	sd1, sd2
Pt3	sd1, sd3
Pt4	sd1, sd2, sd3, sd4

ตารางที่ 7-2 แสดงกลุ่มของ serviceData

7.4.6.1 ค่าเริ่มต้นของ SDE แบบ static ภายในอินเตอร์เฟสของ portType

ค่าเริ่มต้นของ SDE จะสามารถรวมกันลงไปที่อินเตอร์เฟสของ portType ได้ อย่างไรก็ตาม ค่า cardinality ที่ต้องการ (minOccurs และ maxOccurs) จะต้องถูกกันไว้ เช่น ในตัวอย่าง เซอร์วิสอินสแตนซ์ที่สร้าง pt1 จะมีค่า <sd1> 1 </sd1> สำหรับ SDE ที่ชื่อ sd1

```
<gwsdl:portType name="pt1">
   <sd:serviceData name="sd1" minOccurs="1" maxOccurs="1"
                           mutability="static"/>
   <sd:staticServiceDataValues>
      <sd1>1</sd1>
   </ad:staticServiceDataValues>

</gwsdl:portType>
```

ต่อจากนี้จะเป็นการสร้าง pt2 โดยสืบทอดค่า <sd1> 1 </sd1> สำหรับ SDE ที่ชื่อ sd1 และจะมีค่า <sd2> 2 </sd> สำหรับ SDE ที่ชื่อ sd2

```
<gwsdl:portType name="pt2" extends="pt1">
   <sd:serviceData name="sd2" minOccurs="1" maxOccurs="1"
                              mutability="static"/>
   <sd:staticServiceDataValues>
      <sd2>2</sd2>
   </ad:staticServiceDataValues>
</gwsdl:portType>
```

ค่าเซอร์วิสอินสแตนซ์ที่สร้าง pt3 ก็จะมี 2 ค่าคือ <sd3> 3a </sd3> และ <sd3>3b</sd3> สำหรับ SDE ที่ชื่อ sd3 ซึ่งควรจะสืบทอดค่าจาก SDE ที่ชื่อ sd1 ด้วย

```
<gwsdl:portType name="pt3" extends="pt1">
   <sd:serviceData name="sd3" minOccurs="1" maxOccurs="unbounded"
                              mutability="static"/>
   <sd:staticServiceDataValues>
      <sd3>3a</sd3>
      <sd3>3b</sd3>
   </sd:staticServiceDataValues>
</gwsdl:portType>
```

ค่าเซอร์วิสอินสแตนซ์ที่สร้าง pt4 จะสืบทอดจากค่า sd1 ที่กำหนดโดย pt1 แต่การขาดไปของค่า staticServiceDataValues ที่แสดงให้เห็นโดยไม่มีค่าเริ่มต้นสำหรับ sd4 (แม้ว่ากำหนดจาก portType ที่สืบทอดมาจาก pt4)

```
<gwsdl:portType name="pt4" extends="pt1">
   <sd:serviceData name="sd4" minOccurs="0" maxOccurs="unbounded"
                              mutability="static"/>
</gwsdl:portType>
```

เซอร์วิสอินสแตนซ์ที่สร้าง pt5 จะไม่สามารถสร้างได้ ตั้งแต่ที่ไม่มีค่าเริ่มต้นสำหรับ sd5 และตั้งแต่ที่ค่า minOccurs มีค่ามากกว่า 0 โดยจะมีความผิดพลาดเกิดขึ้นและแสดงออกผ่านการสร้างอินสแตนซ์ โดย portType ชนิดนี้จะพบผ่านการกำหนด "abstract" portType ภายใน portTypes ที่มาจากการกำหนดค่าสำหรับ SDEs ด้วย minOccurs ที่มากกว่า 0

```
<gwsdl:portType name="pt5" extends="pt1">
   <sd:serviceData name="sd5" minOccurs="1" maxOccurs="unbounded"
                              mutability="static"/>
</gwsdl:portType>
```

เซอร์วิสอินสแตนซ์ที่สร้าง pt6 ก็ไม่สามารถสร้างได้ ตั้งแต่ที่ portType กำหนดค่าเพิ่มต้นของ SDE ที่ชื่อ sd1 (เรียกใช้ค่าที่สืบเนื่องมาจาก pt1) ซึ่งมีค่ามากกว่า maxOccurs ของ SDE ที่ชื่อ sd1 โดยจะมีความผิดพลาดแจ้งให้ทราบผ่านการสร้างอินสแตนซ์เช่นกัน

```
<gwsdl:portType name="pt6" extends="pt1">
   </sd:staticServiceDataValues>
      <sd1>6</sd1>
   <sd:staticServiceDataValues>
</gwsdl:portType>
```

เซอร์วิสอินสแตนซ์ที่สร้าง pt7 จะมีเซ็ตที่น่าสนใจของ serviceData อยู่ อย่างแรกคือ เป็นค่าเดี่ยว <sd1>1</sd1> สำหรับ SDE ชื่อ sd1 แม้ว่าจะมีการสืบทอด pt1 ผ่าน pt2 และ pt3 ซึ่งค่าเริ่มต้นของ sd1

จะไม่ซ้ำกัน โดยค่า <sd2>2</sd2> เป็นค่าเดียวสำหรับ SDE ที่ชื่อ sd2 ซึ่งสืบทอดมาจาก pt2 และ SDE ที่ชื่อ pt3 จะมีค่า 3 ค่า คือ <sd3>3a</sd3>, <sd3>3b</sd3> (สืบทอดมาจาก pt3) และ <sd3>7</sd3> ซึ่งกำหนดแบบ local นอกจากนี้ยังมีค่าแบบ local ที่กำหนดโดย SDE ที่ชื่อ sd7 คือ <sd7>7</sd7>

```
<gwsdl:portType name="pt7" extends="pt2 pt3">
   <sd:serviceData name="sd7" minOccurs="1" maxOccurs="1"
                              mutability="static"/>
   <sd:staticServiceDataValues>
      <sd7>7</sd7>
      <sd3>7</sd3>
   </sd:staticServiceDataValues>
</gwsdl:portType>
```

โดยสรุปแล้ว ค่า SDE จะรวมอินเตอร์เฟสของ portType ถ้าผลของ SDE ไปกระทบกับ cardinality (ค่าที่น้อยกว่า minOccurs หรือ มากกว่า maxOccurs) ของ SDE แล้ว จะมีการแจ้งความผิดพลาดขึ้นเมื่อเว็บเซอร์วิสนั้นได้ถูกสร้างขึ้น

7.4.7 ค่า serviceData แบบ dynamic

แม้ว่าหลายๆ serviceData จะถูกกำหนดโดยรูปแบบของเซอรีวิสแล้ว บางสถานการณ์ก็อาจจะใช้การเพิ่มหรือการลบ serviceData แบบ dynamic หรือ จากอินสแตนซ์มาช่วย นี่หมายความว่าบางการอัพเดตเป็นจะเป็นการสร้างแบบเจาะจง ตัวอย่างเช่น เซอร์วิสอินสแตนซ์จะมีกระบวนการสร้างค่าเซอร์วิสเดต้าใหม่ๆเพิ่มขึ้นมาได้

โดย GridService portType ที่ใช้กับ SDEs แบบ dynamic นั้น จะประกอบไปด้วยค่า serviceData ที่ชื่อ "serviceDataName" ที่ใช้แสดง serviceData ที่กำหนดในปัจจุบัน คุณสมบัติของอินสแตนซ์นี้จะตอบค่าซุปเปอร์เซ็ตของเซอร์วิสเดต้าที่ประกาศโดย GWSDL กลับไป โดยกำหนดรูปแบบของเซอร์วิส ที่ยอมให้ผู้ร้องขอ ใช้ กระบวนการsubscribe ที่ใช้เปลี่ยน serviceDataSet และ ใช้กระบวนการ findServiceData ที่ใช้เลือกค่าปัจจุบันของ serviceDataSet

7.5 คุณสมบัติสำหรับแก่นของกริดเซอร์วิส

จะแบ่งออกเป็นกลุ่มๆของทุกๆกริดเซอร์วิส ดังนี้

1. Service Description และ Service Instance
2. รูปแบบของเวลาที่ใช้ใน OGSI
3. ค่าไลฟ์ไทม์ของ XML
4. รูปแบบการให้ชื่อและการจัดการการเปลี่ยนแปลง
5. การให้ชื่อกริดเซอร์วิสอินสแตนซ์
6. วัฏจักรชีวิตของกริดเซอร์วิส
7. การจัดการเมื่อเกิดกระบวนการผิดพลาด

8. กระบวนการที่ใช้สืบทอด

7.5.1 Service Description และ Service Instance

ใน OGSI นั้นได้แบ่งแยกระหว่าง description กับ instance ของกริดเซอร์วิส ดังนี้

- Grid Service Description จะเป็นส่วนที่ใช้บ่งบอกว่า client จะตอบโต้กับเซอร์วิสอินสแตนซ์ได้อย่างไร โดยที่ description นี้จะเป็นอิสระต่ออินสแตนซ์อื่นๆ ภายในเอกสารของ WSDL นั้น Grid Service Description จะมาจากการสืบทอดจาก portTypc (เช่น portType ที่อ้างอิงจากค่าพอร์ทลูกของ wsdl:service) ของอินสแตนซ์ พร้อมด้วย portType ที่เกี่ยวข้อง (รวมถึงการประกาศ SDE) , การผูก, ข้อความ และ การกำหนดชนิดต่างๆ

- Grid Service Description ที่ถูกใช้ในเวลาเดียวกันโดย Grid Service Instance นั้น จะมีลักษณะดังต่อไปนี้

1. เกิดเป็นรูปร่างจากที่มาจากการ deseription ของเซอร์วิสที่บ่งบอกว่าจะตอบโต้อย่างไร

2. มี Grid Service Handles อยู่ 1 ค่าหรือ มากกว่านั้น

3. มี Grid Service Reference อยู่ 1 ค่าหรือ มากกว่านั้นที่ใช้อ้างถึง

โดย service description จะใช้ในเบื้องต้นอยู่ 2 อย่าง คือ ใช้เพื่ออธิบายเกี่ยวกับเซอร์วิสซึ่งส่วนนี้จะสามารถใช้เครื่องมือช่วยในการสร้างขึ้นมาได้อย่างอัตโนมัติ และใช้เพื่อค้นหาเซอร์วิส เช่น เพื่อหาเซอร์วิสอินสแตนซ์ที่สร้างขึ้นมาโดยเฉพาะจาก servicc description หรือหา factory ที่สามารถสร้างอินสแตนซ์โดยเฉพาะจาก service description

โดยการบ่งบอกถึง service description นั้น ต้องมี 2 ส่วนคือ syntax และ semantics โดยที่ syntax จะอธิบายได้ด้วย WSDL portType และ semantics จะใช้แสดงผ่านชื่อที่ใช้ใน portType เช่น เมื่อกำหนดกริกเซอร์วิสขึ้นมา โดยใช้ชื่อที่เป็นเอกเทศขึ้นมาหลายๆชื่อ semantic จะพยายามที่จะเกี่ยวข้องกับแต่ละ ชื่อนั้นๆ โดยชื่อเหล่านี้จะสามารถใช้โดย client เพื่อค้นหาเซอร์วิสโดยความต้องการของ scmantics โดยจะค้นหาเซอร์วิสอินสแตนซ์และแฟคทอรี่ด้วยชื่อพิเศษนั้นๆ การใช้ขอบเขตของชื่อเหล่านี้จะเป็นการเตรียมสำหรับการใช้ชื่อที่จะเป็นเอกเทศต่อกัน

7.5.2 รูปแบบของเวลาที่ใช้ใน OGSI

ส่วนที่สำคัญส่วนหนึ่งที่เกิดขึ้นกับการใช้งานของหลายๆกลุ่มในการกระจายของกริดนั้น ก็คือการแสดงเวลานั่นเอง ยกตัวอย่างเช่น มีข้อมูลที่ต้องส่งด้วย timestamps ตามลำดับ เพื่อให้ข้อมูลนั้นจะมีประโยชน์กับผู้ที่นำไปใช้ เพราะบางที client อาจจะจำเป็นต้องใช้เซอร์วิสอินสแตนซ์นั้นในช่วงเวลาที่ยังควรใช้งานอยู่ และหลายๆเซอร์วิสก็ต้องอาศัยความเข้าใจเรื่องลำดับให้ถูกต้องเพื่อให้ client สามารถจัดการและตอบโต้กลับมันได้ทันที

เวลามาตรฐานแบบ GMT นั้น ถูกนำมาใช้ในกริดเซอร์วิส ที่จะยอมให้เวลาโดยรวมกลมกลืนกัน อย่างไรก็ตามการใช้เวลา GMT เป็นมาตรฐานนั้นจะไม่ทำให้เกิดการ synchronization กับทุกๆ

เครื่องได้ เพราะเวลาของแต่ละเครื่องย่อมไม่ตรงกันทั้งหมด ซึ่งส่วนนี้จำเป็นต้องใช้วิธีพิเศษที่จะทำให้เวลานั้น synchronize กันเพื่อคุณภาพของงาน

โดยกริดเซอร์วิสนั้นทั้งฝั่ง hosting environments และฝั่ง client จะต้องใช้เวลาร่วมกันผ่าน Network Time Protocol (NTP) หรือ อาจจะใช้ฟังก์ชั่นพิเศษที่จะทำให้เวลานั้นตรงตาม GMT อยู่เสมอ อย่างไรก็ตาม client และ service ต้องมีการยอมรับข้อความที่บรรจุค่าเกี่ยวกับเวลาไว้แล้วเกินขอบเขตบ้าง เพราะ การ synchronize นั้นไม่เพียงพอ ซึ่งอาจทำให้ข้อมูลที่สำคัญส่งไม่ถึงได้

ในบางกรณีก็อาจจะต้องการเวลาที่เป็น 0 หรือเป็นอนันต์ก็มี โดยเวลาที่เป็น 0 นั้นจะใช้แสดงแทนเวลาที่ผ่านไปแล้วในอดีต ส่วนเวลาที่เป็นค่าอนันต์จะใช้สำหรับแสดงความคิดเกี่ยวกับเวลา โดยในขอบเขตของชื่อใน ogsi นั้นจะแทนที่ค่าเวลาพิเศษนั้นๆใน xsd:dateTime ดังตัวอย่าง

```
…  targetNamespace =
     "http://www.gridforum.org/namespaces/2003/03/OGSI"

<simpleType name="ExtendedDateTimeType">
   <union memberTypes="ogsi:InfinityType xsd:dateTime"/>
</simpleType>

<simpleType name="InfinityType">
   <restriction base="string">
      <enumeration value="infinity"/>
   </restriction>
</simpleType>
```

7.5.3 ค่าไลฟ์ไทม์ของ XML

ตั้งแต่ที่ serviceData จะต้องแทนไดนามิคสเตทของเซอร์วิสอินสแตนซ์ จึงเป็นเรื่องจำเป็นที่จะต้องเข้าใจเกี่ยวกับไลฟ์ไทม์ที่ถูกต้อง โดยไคล์เอนท์จะต้องใช้ข้อมูลที่เกี่ยวพันกับเวลาสำหรับการใช้ในเรื่องต่างๆ จนกระทั่งถึงการอิสระที่จะปฏิเสธข้อมูลนั้นๆด้วย

ซึ่งจะใช้ 3 แอททริบิวท์ของ XML ที่เป็นข้อมูลที่เกี่ยวข้องกับค่าไลฟ์ไทม์ คือ

- ogsi:goodFrom: ใช้กำหนดเวลาจากที่ซึ่งข้อมูลถูกต้อง ส่วนนี้ใช้เวลาตอนที่เริ่มสร้างมาใช้

- ogsi:goodUntil: ใช้กำหนดเวลาจนกระทั่งข้อมูลยังถูกต้อง ถ้าใช้ส่วนนี้จะมีคุณสมบัติที่ดีกว่าหรืออย่างน้อยเทียบเท่ากับแบบ goodFrom

- ogsi:availableUntil: ใช้กำหนดเวลาจนกระทั่งถึงช่วงที่ค่าที่คาดหวังยังคงใช้ได้ บางทีอาจจะมีการอัพเดตค่านั้นด้วย ซึ่งเวลาแบบนี้นั้นไคล์เอนท์จะต้องได้รับค่าที่อัพเดตนี้ด้วย หลังจากนั้นแล้วไคล์เอนท์ ไม่ควรที่จะได้รับค่านี้อีกที เนื่องจากอาจมีปัญหากับกฎหรือสิ่งที่สำคัญอื่นๆ คุณสมบัติแบบนี้ต้องดีกว่าหรือเทียบเท่ากับแบบ goodFrom

ตัวอย่างการใช้ XML สำหรับแอททริบิวท์เหล่านั้น

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:attribute name="goodFrom" type="ogsi:ExtendedDateTimeType"/>
<xsd:attribute name="goodUntil" type="ogsi:ExtendedDateTimeType"/>
<xsd:attribute name="availableUntil" type="ogsi:ExtendedDateTimeType"/>

<xsd:attributeGroup name="LifeTimePropertiesGroup">
   <xsd:attribute ref="ogsi:goodFrom" use="optional"/>
   <xsd:attribute ref="ogsi:goodUntil" use="optional"/>
   <xsd:attribute ref="ogsi:availableUntil" use="optional"/>
</xsd:attributeGroup>
```

ซึ่งจะใช้ serviceData ค่านี้เพิ่มกำหนดไลฟ์ไทม์ตามนี้

```
<wsdl:definitions
   targetNamespace="http://example.com/ns"
   xmlns:n1="http://example.com/ns"
   … >

   <wsdl:types>
      <xsd:schema …
         "targetNamespace=http://example.com/ns"
         …>
         <xsd:complexType name="MyType">
            <xsd:sequence>
               <xsd:element name="e1" type="xsd:string" minOccurs="1"/>
               <xsd:element name="e2" type="xsd:string" minOccurs="1"/>
               <xsd:element name="e3" type="xsd:string" minOccurs="0"/>
            </xsd:sequence>
         <anyAttribute namespace="##any"/>
         </xsd:complexType>
      </xsd:schema>
   </wsdl:types>
   …
   <gwsdl:portType name="MyPortType">
      …
      <sd:serviceData name="mySDE" type= "n1:MyType"
                      minOccurs="1" maxOccurs="1"
                      mutability="mutable"/>
      …
   </gwsdl:portType>
   …
</wsdl:definitions>
```

และภายในเซอร์วิสอินสแตนซ์นั้นจะต้องมีค่า serviceDataValues ดังนี้

```
<sd:serviceDataValues
   <n1:mySDE
         goodFrom="2002-04-27T10:20:00.000-06:00"
         goodUntil="2002-04-27T11:20:00.000-06:00"
         availableUntil="2002-04-28T10:20:00.000-06:00">
      <n1:e1>
         abc
      </n1:e1>
      <n1:e2 ogsi:goodUntil="2002-04-27T10:30:00.000-06:00">
         def
      </n1:e2>
      <n1:e3 ogsi:availableUntil="2002-04-27T20:20:00.000-06:00">
         ghi
      </n1:e3>
   </n1:mySDE>
</sd:serviceDataValues>
```

โดยการใช้ goodFrom และ goodUntil ของ n1:mySDE นั้นจะอ้างถึงค่าลูกทั้งหมด ที่แอททริบิวท์เหล่านั้นกำหนดให้ใน SDE เป็นค่าไลฟ์ไทม์ที่คาดหวังไว้สำหรับค่านั้น ซึ่งในตัวอย่างนี้จะใช้จาก 10.20am ไปจนถึง 11.20 am ในวันที่ 27 เดือนเมษายน ปี 2002 โดยผู้ที่นำไปใช้จะได้รับ SDE นี้ที่เวลา 10.20am ดังนั้นแม้ว่าผู้ที่ใช้งานจะได้รับช้ากว่านั้นอีก 1 ชม. ก็จะสามารถใช้งานได้ถูกต้อง โดยที่ไคล์เอนท์บางทีอาจจะคิวรี่เซอร์วิสอินสแตนซ์นี้อีกครั้งเมื่อเวลา 11.20 เพื่อรับค่าใหม่ก็ได้

7.5.4 รูปแบบการให้ชื่อและการจัดการการเปลี่ยนแปลง

ส่วนสำคัญของระบบแบบกระจายนั้นก็คือต้องยอมรับค่าที่อ่จเปลี่ยนแปลงตลอดเวลาได้ เพื่อให้ได้สิ่งนี้แล้ว client จึงต้องการที่จะเลือกเซอรีวิสที่เปลี่ยนแปลงไปได้เพื่อที่จะนำใช้ในการทำงานของเขา ส่วนนี้จะเป็นส่วนที่แสดงถึงการจัดการส่วนนี้ของ OGSI

7.5.4.1 ปัญหาของการจัดการการเปลี่ยนแปลง

ในกริดเซอร์วิสอินสแตนซ์นั้นจะมีการจำกัดความได้อยู่ 2 ลักษณะคือ

1. เป็นรูปแบบคุณสมบัติของมัน ซึ่งแบบนี้ กริดเซอร์วิสอินสแตนซ์จะกำหนดโดย service description ซึ่งประกอบไปด้วย portTypes, operations, serviceData, messages และ types

2. เป็นรูปแบบการทำงาน ส่วนนี้อาจจะแสดงมาจากแบบแรกด้วย โดยจะบอกว่าแท้จริงแล้วมันให้กริดเซอร์วิสอินสแตนซ์ได้อย่างไร ซึ่งรูปแบบการทำหรือความผิดพลาดจะแสดงผลออกมาที่เซอร์วิสอินสแตนซ์

สำหรับไคล์เอนท์เพื่อที่จะค้นหาและใช้งานกริดเซอร์วิสอินสแตนซ์ได้อย่างถูกต้องนั้นไคล์เอนท์ จะต้องเลือกจากคำจำกัดความของ 2 ชนิดที่กล่าวมา แต่เมื่อลองพิจารณาดูการที่ Grid service description บ่งบอกค่าออกมาเป็น portType นั้นเป็นสิ่งที่ไคล์เอนท์ต้องการจริงมั้ย? และก็ส่วนที่เป็นโครงของการทำงานนั้นจะไม่มีความผิดพลาดจนต้องแก้ไขเลยหรือ?

ต่อมา grid service description นั้นจำเป็นจะต้องมีออกมาอยู่ตลอดเวลา ถ้ากริดเซอร์วิสอินสแตนซ์นั้นสืบทอดมาจากส่วนข้างหลัง แล้วไคล์เอนท์ต้องการส่วนนั้นก็ควรต้องรองรับส่วนที่มันเป็น

ส่วนที่สืบทอดมาได้ด้วย เช่น เพิ่ม operation หรือ service description ใหม่ๆขึ้นมา ซึ่งบางทีอาจเกิดจากการเปลี่ยนแปลงที่จำเป็น เช่น แก้ไขบั๊ก, ทำไว้ให้แก้ไขความผิดพลาดที่เกิดขึ้น เป็นต้น

อย่างไรก็ตามไคล์เอนท์ ต้องสามารถค้นพบ grid service description ที่เปลี่ยนแปลงที่ไม่ใช่ backward-compatible ได้ ซึ่ง backward-compatible นั้นจะทำให้เกิดการเปลี่ยนรูปแบบหรือการทำของกริดเซอร์วิสได้

7.5.4.2 ระเบียบการให้ชื่อของ Grid Service Description

ใน WSDL แต่ละ portType จะใช้ถูกใช้ทั่วไปและมีชื่อที่เป็นเอกเทศกันผ่านเงื่อนไขการมีชื่อของมัน ซึ่งจะมีการรวมกันของขอบเขตของชื่อที่บอกด้วย portType และมีค่าชื่อที่เป็นเอกเทศกันในแต่ละ ค่า portType ภายในขอบเขตของชื่อนั้นๆ ใน OGSI จะใช้การควบคุมการจัดการโดยอาศัยค่าทั้งหมดของ grid service description ที่เปลี่ยนแปลงไม่ได้ โดยใช้ QName ของ Grid service prtType, operation, message, serviceData และ underlying type จะต้องกำหนดจาก WSDL/XSD ถ้าเปลี่ยนค่านี้ไปแล้ว portType ใหม่นั้นจะต้องกำหนดค่า QName ใหม่ด้วย นั้นหมายความว่าจะมีชื่อกายในใหม่ และมีขอบเขตชื่อใหม่ด้วย

ระหว่างพัฒนาระบบนั้น grid service instance จะสามารถเปลี่ยนแปลงได้ทั้งรูปแบบ, portType หรือแม้กระทั่งระดับการ implement ทำให้ผู้พัฒนาระบบต้องเลือกที่จะใช้งานมันอย่างระมัดระวัง ซึ่งก็ไม่น่ากังวลนักเพราะว่าจะการเปลี่ยนแปลงจะต้องผ่านการแสดงค่าของ portType ใหม่ๆนั้นอยู่แล้ว

7.5.5 การให้ชื่อกริดเซอร์วิสอินสแตนซ์

แต่ละกริดเซอร์วิสอินสแตนซ์นั้นจะมี อย่างน้อย 1 Grid Service Handles (GSH) ดังที่กล่าวไปช่วงต้น ซึ่ง GHS นั้นจะใช้ชื่อที่อยู่ในรูปแบบของ URI และไม่ได้เป็นส่วนที่เก็บข้อมูลว่าจะยอมให้ client ติดต่อตรงมาที่เซอร์วิสอินสแตนซ์อย่างไร โดยที่ client นั้นจะทราบว่าจะต้องติดต่อตรงกับเซอร์วิสอินสแตนซ์ไหนนั้นจะต้องผ่านการ resolve ค่า GSH ให้เป็น Grid Service Reference (GSR) ก่อน โดยที่ GSR จะเป็นส่วนที่เก็บข้อมูลที่จำเป็นที่จะใช้ในการติดต่อกับเซอร์วิสอินสแตนซ์ผ่านการผูกของโปรโตคอลในเครือข่ายนั้นๆ

เหมือนกับ URI โดย GSH จะเกิดจาก scheme โดยตามด้วยสตริงที่บรรจุข้อมูลพิเศษ โดย scheme นั้นจะแสดงว่ามันจะแสดงค่าอะไรบ้าง และ resolve จาก GSH ให้เป็น GSR ได้ยังไง โดย client จะเลือกที่จะ resolve GSH ด้วยตนเองหรืออาจจะเลือกโดยอาศัยแหล่งภายนอกก็ได้ เช่น แก้ไขวิธีการทำ HandleResolver ที่ portType เป็นต้น

รูปแบบของ GSR จะเป็นลักษณะพิเศษโดยใช้ผ่าน client เพื่อติดต่อสื่อสารกับกริดเซอร์วิสอินสแตนซ์ เช่น RMI/IIOP จะใช้มัน FSR จะเลือกรูปแบบของ IOR แต่ถ้าใช้ SOAP แล้ว GSR ก็จะเลือกรูปแบบของ WSDL มาใช้

ขณะที่ GSH จะถูกต้องในไลฟ์ไทม์ของกริดเซอร์วิสอินสแตนซ์นั้น ถ้า GSR มีการผิดพลาดจะทำให้ไคล์เอนท์ต้องการที่จะ resolve ใหม่เพื่อให้ได้ค่า GSR ที่ถูกต้อง

7.5.5.1 Grid Service Reference (GSR)

กริดเซอร์วิสอินสแตนซ์นั้นจะสร้างการเข้าถึงกับแอพพลิเคชั่นของ client ได้ผ่านการใช้ GSR โดยที่ GSR จะเป็น network-wide pointer เพื่อเข้าสู่กริดเซอร์วิสอินสแตนซ์ที่อยู่ในโฮสต์ที่พร้อมจะทำงาน โดยแอพพลิเคชั่นของ client นั้นจะสามารถใช้ GSR เพื่อส่งการร้องขอตรงไปที่อินสแตนซ์ที่กำหนดโดย GSR ซึ่ง GSR จะรองรับการเขียนโปรแกรมผ่านกริดเซอร์วิสอินสแตนซ์ที่ชื่อ "by reference" ซึ่ง GSR จะบรรจุข้อมูลที่จำเป็นสำหรับการที่จะเข้าถึงกริดเซอร์วิสอินสแตนซ์ในโฮสต์ได้ผ่านหลายๆโปรโตคอล

การเอนโค้ดของ GSR นั้น บางทีจะได้รับจากหลายๆรูปแบบในระบบ เหมือนกับกระบวนการอื่นๆในระบบ โดยการเอนโค้ดที่แท้จริงนั้นจะผ่านรูปแบบของ GSR ที่เปรียบเสมือนการผูกผ่านเว็บเซอร์วิสด้วยกริดเซอร์วิสอินสแตนซ์ อย่างที่กำหนดไว้ในรูปแบบการเอนโค้ด WSDL ของ GSR ที่ใช้สำหรับการผูกในบางครั้ง

GSR จะใช้รูปแบบ "on the wire" จากการสร้าง GSR ในกริดเซอร์วิสโฮสต์ เมื่อส่วนที่ใช้อ้างได้ส่งไปที่พารามิเตอร์ที่เป็นของกระบวนการที่ใช้กำหนด WSDL แล้ว แอพพลิเคชั่นของไคล์เอนท์จะส่งผ่านการร้องขอข้อความจาก WSDL-defined operation ซึ่งควรจะรวมทั้งหมดของที่อยู่ที่มีเซอร์วิสที่ต้องการจะสื่อสารนั้นผ่านการสร้างเซอร์วิสอินสแตนซ์ด้วย

ค่าใดๆของ GSR นั้นจะต้องคงอยู่ในระบบ โดยที่วัฏจักรชีวิตของ GSR จะต้องเป็นอิสระกับความเกี่ยวข้องของกริดเซอร์วิสอินสแตนซ์ ซึ่ง GSR จะยังคงถูกต้องตราบเท่าที่ เซอร์วิสอินสแตนซ์ยังคงสามารถใช้ได้ผ่าน GSR นี้

เมื่อ GSR ถูกพบว่าถูกต้องและได้ถูกสร้างจากกริดเซอร์วิสอินสแตนซ์ที่ยังคงอยู่ client จะได้รับค่า GSR ใหม่นั้นผ่านการใช้ GSH ของกริดเซอร์วิสอินสแตนซ์ที่เกี่ยวข้อง โดยที่ GSH จะบรรจุค่า binding-specific ที่ใช้สร้าง GSR ไว้ โดยที่ binding-specific ของ GSR นั้นจะรวมทั้ง เวลาที่หมดอายุด้วยที่จะประกาศเมื่อ client ต้องการ GSR นั้น ซึ่ง GSR จะต้องถูกต้องตลอดถึงช่วงเวลานั้นและควรจะผิดพลาดเมื่อเลยค่าเวลานั้น

ตัวอย่างโครง XML ที่ใช้กำหนด GSR ตามนี้

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:element name="reference" type="ogsi:ReferenceType"/>

<xsd:complexType name="ReferenceType" abstract="true">
   <xsd:attribute ref="ogsi:goodFrom" use="optional"/>
   <xsd:attribute ref="ogsi:goodUntil" use="optional"/>
</xsd:complexType>
```

7.5.5.1.1 WSDL ที่ใช้เอนโค้ด GSR

เป็นสิ่งที่จำเป็นที่ WSDL จะต้องเอนโค้ด GSR ที่บรรจุข้อมูลที่จำเป็นน้อยที่สุดสำหรับว่ามัน

จะต้องถึงกริดเซอร์วิสอินสแตนซ์นั้นได้อย่างไร โดย WSDL GSR จะมีค่า wsdl:definition ซึ่งบรรจุค่า wsdl:service ไว้และอาจจะมีค่าอื่นๆด้วย ดูได้จากโค้ดนี้

```
targetNamespace = http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:complexType name="WSDLReferenceType">
   <xsd:complexContent>
      <xsd:extension base="ogsi:ReferenceType">
         <xsd:sequence>
            <xsd:any namespace="http://schemas.xmlsoap.org/wsdl/"
                 minOccurs="1" maxOccurs="1" processContents="lax"/>
         </xsd:sequence>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
```

7.5.5.2 Grid Service Handles (GSH)

ต้องมีคุณสมบัติตามนี้

1. GSH ต้องมีค่า URI ที่ถูกต้อง
2. GSH จะต้องใช้ได้อย่างทั่วถึงเมื่อเกิดจากการอ้างกริดเซอร์วิสอินสแตนซ์เดียวกัน และ GSH ต้องไม่อ้างมากกว่า 1 กริดเซอร์วิสอินสแตนซ์
3. กริดเซอร์วิสอินสแตนซ์นั้นจะต้องมีอย่างน้อย 1 GSH
4. กริดเซอร์วิสอินสแตนซ์อาจจะมีหลายๆ GSH ได้ถ้าใช้ URI ที่ไม่เหมือนกัน
5. กริดเซอร์วิสอินสแตนซ์จะต้องไม่มีหลายๆ GSH ถ้าใช้ URI ที่เหมือนกัน
6. ค่าของ gridServiceHandle ของกริดเซอร์วิสอินสแตนซ์จะต้องมีค่าเฉพาะค่า SGH ที่อ้างถึงอินสแตนซ์นั้นเท่านั้น
7. อาจจะมีหลายๆ GSRs ที่อ้างไปที่กริดเซอร์วิสอินสแตนซ์อันเดียวกัน
8. ผลจาก GSH อันเดียวกัน อาจทำให้เกิด GSR ที่ต่างกันได้ โดย resolver จะตอบค่า GSR ที่ต่างกันผ่านค่า GSH เดียวกันที่เวลาต่างกัน และอาจจะส่งค่า GSR ที่ต่างกันไปให้ client ที่ต่างกันด้วย
9. GSH อาจจะไม่สามารถ resolve GSR ได้ ไม่ว่าจะก่อน,ในระหว่าง หรือหลังจากที่กริดเซอร์วิสอินสแตนซ์นั้นยังคงมีชีวิตอยู่ อย่างไรก็ตามโอกาสแบบนี้จะเกิดขึ้นไม่บ่อยนัก เนื่องจากรูปแบบของ URI ที่ใช้จะมีคุณภาพที่ค่อนข้างดีพอ
10. ผลจากการ resolve จาก GSH เป็น GSR นั้น บางทีอาจจะเชื่อถือได้หรืออาจเชื่อถือไม่ได้ก็เป็นได้ ขึ้นอยู่กับหลายๆอย่าง เช่น ค่าต่างๆที่ client เซ็ตไว้ หรือการจำกัดของโปรโตคอลที่ใช้ resolve

ตัวอย่าง XML ที่ใช้กำหนด GSH

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:element name="handle" type="ogsi:HandleType"/>

<xsd:simpleType name="HandleType">
   <xsd:restriction base="xsd:anyURI"/>
</xsd:simpleType>
```

7.5.5.3 เซอร์วิสโลเคเตอร์

เซอร์วิสโลเคเตอร์นั้นเป็นโครงสร้างที่บรรจุค่า GSH, ค่า GSR และค่าอินเตอร์เฟสของ QNames (portType) ไว้อย่างน้อย 0 หรือมากกว่านั้น โดยค่า GSHs และ GSRs ทั้งหมดนั้นจะอ้างอิงไปที่กริดเซอร์วิสอินสแตนซ์เดียวกัน และอินเตอร์เฟสทั้งหมดนั้นต้องถูกพัฒนาขึ้นจากกริดเซอร์วิสอินสแตนซ์นั้น โดยโลเคเตอร์จะใช้เพื่อยอมรับ GSH หรือ GSR นั้น โดยการกำหนด XML สำหรับโลเคเตอร์ทำตามนี้

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:element name="locator" type="ogsi:LocatorType"/>

<xsd:complexType name="LocatorType">
   <xsd:sequence>
      <xsd:element ref="ogsi:handle"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="ogsi:reference"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="interface" type="QName"
                   minOccurs="0" maxOccurs="unbounded"/>
   </xsd:sequence>
</xsd:complexType>
```

7.5.6 วัฏจักรชีวิตของกริดเซอร์วิส

วัฏจักรชีวิตของทุกๆกริดเซอร์วิสอินสแตนซ์นั้นจะกำหนดโดยการสร้างและการทำลายของเซอร์วิสอินสแตนซ์นั้นๆ โดยวิธีที่แท้จริงของกริดเซอร์วิสอินสแตนซ์จะถูกสร้างหรือถูกทำลายโดยคุณสมบัติขั้นพื้นฐานของกลุ่มโฮสต์นั้นๆ แต่กระนั้นก็ยังมีกลุ่มของ portTypes ที่เกี่ยวข้องกันได้กำหนดว่าไคล์เอนท์จะติดต่อกับเหตุการณ์ในวัฏจักรชีวิตนี้ได้อย่างไร โดยจะแบ่งเป็นเหตุการณ์ย่อยๆดังนี้

- ไคล์เอนท์จะร้องขอเพื่อสร้าง (creation) กริดเซอร์วิสอินสแตนซ์โดยการเรียก createService ของเซอร์วิสอินสแตนซ์ที่จะสร้างขึ้นโดย Factory portType หรือ จากวิธีพิเศษที่ใช้กำหนด เซอร์วิสใหม่ๆ เช่น NotificationSource portType ซึ่งเป็นเซอร์วิสของ factory ที่จะกล่าวถึงต่อไป
- ไคล์เอนท์จะร้องขอเพื่อทำลาย (destruction) กริดเซอร์วิสอินสแตนซ์ ผ่านไคล์เอนท์ใดๆ โดย explicit destruction operation ที่รองรับโดย GridService portType ซึ่งไคล์เอนท์ที่ลงทะเบียนนั้นจะสนใจกับกริดเซอร์วิสอินสแตนซ์เพียงช่วงเวลาหนึ่ง จนกระทั่งเวลานั้นหมดไป โดยที่เซอร์วิสอินสแตนซ์นั้นจะต้องถูกทำลายโดยอัตโนมัติ

กริดเซอร์วิสอินสแตนซ์จะรองรับการจัดการไลฟ์ไทม์แบบซอฟท์สเตท (Soft state) ในกรณีที่ไคล์เอนท์จะเริ่มค่าไลฟ์ไทม์นั้นเมื่อกริดเซอร์วิสอินสแตนซ์ได้ถูกสร้างขึ้นผ่าน factory และได้รับการยืนยันจากข้อความ requestTerminationBefore/After ("keepalive") เพื่อร้องขอที่จะรับไลฟ์ไทม์ของเซอร์วิสนั้น ถ้าถึงเวลาสิ้นสุดแล้ว ฝั่งเซิร์ฟเวอร์ที่เก็บเซอร์วิสอินสแตนซ์นั้นก็จะทำลาย และรีซอร์สที่เกี่ยวข้องทั้งหมดทิ้งไป

ซึ่งเวลาสิ้นสุดนั้นจะเปลี่ยนอย่างไม่ซ้ำกัน การที่ไคล์เอนท์ร้องขอเวลาสิ้นสุดนั้นจะเร็วกว่าขอเวลาในตอนปัจจุบัน นั่นคือ ถ้ามีการร้องขอเวลาสิ้นสุดโดยที่เวลานั้นเป็นก่อนเวลาปัจจุบัน นั่นคือจำเป็นต้องแสดง สิทธิเพื่อยกเลิกในครั้งนั้นๆเป็นพิเศษ
เวลาสิ้นสุดที่แสดงผ่าน OGSI ดูได้ ดังตัวอย่างนี้

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:complexType name="TerminationTimeType">
   <xsd:attribute name="after" type="ogsi:ExtendedDateTimeType"
                  use="optional"/>
   <xsd:attribute name="before" type="ogsi:ExtendedDateTimeType"
                  use="optional"/>
   <xsd:attribute name="timestamp" type="xsd:dateTime"
                  use="optional"/>
</xsd:complexType>

<xsd:simpleType name="ExtendedDateTimeType">
   <xsd:union memberTypes="ogsi:InfinityType xsd:dateTime"/>
</xsd:simpleType>

<xsd:simpleType name="InfinityType">
   <xsd:restriction base="string">
      <xsd:enumeration value="infinity"/>
   </xsd:restriction>
</xsd:simpleType>
```

โดยค่า after ของ TerminationTimeType นั้นจะบรรจุเวลาแรกสุดที่เซอร์วิสอินสแตนซ์วางไว้ว่าจะยกเลิก โดยมีค่าพิเศษคือ ค่าอนันต์ ที่ทำให้เซอร์วิสอินสแตนซ์นั้นจะคงอยู่อย่างไม่มีกำหนด

โดยค่า before ของ TerminationTimeType นั้นจะบรรจุเวลาสุดท้ายที่เซอร์วิสอินสแตนซ์วางไว้ว่าจะยกเลิก โดยมีค่าพิเศษคือ ค่าอนันต์ ที่ทำให้เซอร์วิสอินสแตนซ์นั้นจะคงอยู่โดยไม่มีการยกเลิก

โดยค่า timestamp ของ TerminationTimeType จะบรรจุเวลาที่เซอร์วิสอินสแตนซ์ได้เลือกไว้ว่าเวลาที่หลังจากและก่อนหน้าที่กำหนดไว้จะยังคงถูกต้องอยู่ โดย timestamp นั้นจะใช้ใน TerminationTimeType เพื่อใช้เป็นเกณฑ์ของเวลาที่จะเกี่ยวข้องกับ client หรือแหล่งเวลาอื่นๆ

7.5.7 การจัดการเมื่อเกิดกระบวนการผิดพลาด

OGSI จะกำหนด XSD ไว้สำหรับความผิดพลาดทั้งหมดที่กริดเซอร์วิสส่งกลับมาเพื่อใช้แก้ไขปัญหาต่างๆ โดยบรรจุข้อความที่ผิดพลาดต่างๆไว้ โดย ogsi:FaultTypc XSD ดังนี้

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:element name="fault" type="FaultType">

<xsd:complexType name="FaultType">
   <xsd:sequence>
      <xsd:element name="description"
                   type="xsd:string"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="originator"
                   type="ogsi:LocatorType"
                   minOccurs="1" maxOccurs="1"/>
      <xsd:element name="timestamp"
                   type="xsd:dateTime"
                   minOccurs="1" maxOccurs="1"/>
      <xsd:element name="faultcause"
                   type="ogsi:FaultType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="faultcode"
                   type="ogsi:FaultCodeType"
                   minOccurs="0" maxOccurs="1"/>
      <xsd:element name="extension"
                   type="ogsi:ExtensibilityType"
                   minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="FaultCodeType">
   <xsd:simpleContent>
      <xsd:extension base="xsd:string">
         <xsd:attribute name="faultscheme" type="anyURI"
                        use="required"/>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ExtensibilityType">
   <xsd:sequence>
      <xsd:any namespace="##any"/>
   </xsd:sequence>
</xsd:complexType>
```

โดยค่าความผิดพลาดทั้งหมดจากกริดเซอร์วิสอินสแตนซ์นั้นจะใช้ FaultType ที่แก้ไขต่างๆกัน โดยค่าต่างๆที่ใช้ใน OGSI portTypes จะใช้ดังนี้

ค่า description จะบรรจุรายละเอียดความผิดพลาดเป็นภาษาตรงๆ กล่าวคือ เป็นส่วนอธิบายความผิดพลาดกับผู้ใช้งาน ค่านี้เป็นแบบ optional ไม่จำเป็นต้องมีก็ได้

ค่า originator เป็นตัวบอกตำแหน่งของเซอร์วิสอินสแตนซ์ที่ผิดพลาด

ค่า timestamp เป็นค่าเวลาที่ความผิดพลาดเกิดขึ้น

ค่า faultcause เป็นค่าใน ogsi:FaultType ใช้อธิบายผลลัพธ์ที่ผิดพลาด ใช้ร่วมกับ xsi:type ใช้เพื่อให้ทราบรายละเอียดเหตุผลของความผิดพลาด

ค่า faultcode ใช้ในการรายงานความผิดพลาดให้ระบบ โดยอาศัย faultscheme ที่แสดงเป็น URI

ค่า extension ใช้ระบุข้อความพิเศษที่เกี่ยวกับความผิดพลาดในรูปแบบขอบเขตชื่อของ XML ซึ่งบางทีจะเป็นค่าตัวเลขของค่าที่สืบเนื่องมา

โดยค่าความผิดพลาดทั้งหมดนั้นจะตอบกลับในรูป ogsi:fault ในการแจ้งความผิดพลาดดังนี้

```
<wsdl:definitions …>
   <types>
      <xsd:schema …>
         <xsd:complexType name="MyFaultType">
            <xsd:complexContent>
               <xsd:extension base="ogsi:FaultType/>
            </xsd:complexContent>
         </xsd:complexType>
         <xsd:element name="myFault" type="tns:MyFaultType"/>
      </xsd:schema>
   </types>

   <message name=myFaultMessage">
      <part name="fault" element="tns:myFault"/>
   </message>

   <gwsdl:portType …>
      <wsdl:operation …>
         <input …>
         <output …>
         <fault name="myFault" message="tns:myFaultMessage"/>
         <fault name="fault" message="ogsi:faultMessage"/>
      </wsdl:operation>
    </gwsdl:portType>
</wsdl:definitions>
```

7.5.8 กระบวนการที่ใช้ขยาย

หลายๆกระบวนการใน OGSI จะรับค่าที่ไม่มีรูปแบบสืบทอดกันมา เช่น รับค่าที่ให้ไคล์เอนท์ค้นหากระบวนการที่ถูกต้องเป็นต้น ดูได้จากส่วนที่เป็น NotificationSource::subscribe ที่ใช้เพื่อให้ไคล์เอนท์ถามหาเซอร์วิสอินสแตนซ์ที่ต้องการใช้ ของ serviceDataValues ที่เปลี่ยน อย่างไรก็ตามเซอร์วิสที่ใช้ทำ portType ที่สืบทอดมาจาก NotificationSource นั้นจะแสดงกำหนดโดยรูปแบบการคิวรี่ซึ่งมีประสิทธิภาพมากกว่า โดยส่วนนี้จะอธิบายถึงไคล์เอนท์จะเลือกการลงท้ายจาก NotificationSource portType ได้อย่างไร ดูได้จากตัวอย่างข้างล่างนี้

```
targetNamespace = "http://www.gridforum.org/namespaces/2003/03/OGSI"

<xsd:complexType name="OperationExtensibilityType">
  <xsd:attribute name="inputElement" type="QName" use="optional"/>
</xsd:complexType>
```

แต่ละ portType จะมีการกำหนด serviceData ของ OperationExtensibilityType และมีค่า mutability= “static” โดยค่า static ของ SDE นี้จะกำหนดด้วยค่าที่สืบทอดอย่างถูกต้อง ดังตัวอย่าง เช่น มี portType ชื่อ myPT ซึ่งมี myOperation อยู่ ซึ่งจะสร้างวิธีเรียกค่าอนพุทได้ 2 แบบ คือ

myop:myOption1 และ myop:myOption2 ที่ผ่านเข้าไปสู่ส่วนที่สืบทอดจาก myOperation ได้ โดยการกำหนด myOperation ทำได้ตามนี้

```
<gwsdl:portType name="myPT">
   …
   <sd:serviceData name="myOperationExtensiblity"
           type="ogsi:OperationExtensiblityType"
           minOccurs=0 maxOccurs="unbounded"
           mutability="static"
           modifiability="false"
           nillable="false" />
   …
   <sd:staticServiceDataValues>
      <myOperationExtensibility inputElement="m1:myOption1"/>
      <myOperationExtensibility inputElement="m1:myOption2"/>
   </sd:staticServiceDataValues>
   …
</gwsdl:portType>
```

โดยค่า inputElement ของ SDE จะเป็นค่า QName ที่เป็นเอกเทศกัน โดยมาจากการประกาศของค่า XSD ที่กำหนดที่ค่าที่ถูกต้อง โดยคุณสมบัติทั้งหมดจะถูกแสดงโดย inputElement ของ OperationExtensibilityType เช่น inputElement ที่แสดงโดยค่าพารามิเตอร์จากกระบวนการที่แสดงว่ากระบวนนั้นจะรับค่า inputElement นั้นได้อย่างไร

ถ้า inputElement นั้นถูกละเว้นจากค่า SDE แล้ว มันจะต้องถูกแสดงด้วยการขยายค่าที่ผ่านการกระบวนการเรียกนั้น

ตัวอย่างกระบวนการที่รวมค่า portTypes ที่ขยายจากการประกาศ serviceData เช่น มี portType ชื่อ myPT2 ที่ขยาย myPT จะกำหนดค่าที่ถูกต้องให้กับ myOperation ตามนี้

```
<gwsdl:portType name="myPT2" extends="m1:myPT>
   …
   <sd:staticServiceDataValues>
      <myOperationExtensibility inputElement="m2:myOption3"/>
      <myOperationExtensibility/>
   </sd:staticServiceDataValues>
   …
</gwsdl:portType>
```

ในตัวอย่างนี้ เซอร์วิสอินสแตนซ์ที่ใช้ทำ myPT2 จะรองรับการขยายอินพุทของ myOperation: m1:myOption1, m1:myOption2, m2:myOption3 และ ไม่มีค่าที่ทั้งหมด โดยแต่ละกระบวนการจะเกี่ยวข้องกันกับ inputElement

7.6 อินเตอร์เฟสของกริดเซอร์วิส

ส่วนนี้เป็นส่วนที่ระบุเกี่ยวกับการรวมตัวของ WSDL portTypes และพฤติกรรมที่เกี่ยวข้อง

ทั้งหมด โดยเป็นกลุ่มฟังก์ชั่นพื้นฐานของแนวทางการคิดแบบกระจาย โดยจะอธิบายในตารางด้านล่าง และจะแยกออกเป็นหัวข้อย่อย ในส่วนถัดไป

portType Name	Description
GridService	เป็นส่วนที่แสดงรูปแบบของเซอร์วิสนั้นๆ
HandleResolver	ทำการโยงจาก GSH ให้เป็น GSR
NotificationSource	ยอมให้ client ลงท้ายข้อความประกาศ
NotificationSubscription	กำหนดความสัมพันธ์ระหว่างคู่ NotificationSource กับ NotificationSink
NotificationSink	กำหนดกระบวนการส่งข้อความประกาศไปที่เซอร์วิสอินสแตนซ์
Factory	เป็นกระบวนการมาตรฐานที่ใช้สร้าง กริดเซอร์วิสอินสแตนซ์
ServiceGroup	ยอมให้ client ดูแลกลุ่มของเซอร์วิส
ServiceGroupRegistration	ยอมให้กริดเซอร์วิสเพิ่มหรือลดออกจาก ServiceGroup
ServiceGroupEntry	กำหนดความสัมพันธ์ระหว่างกริดเซอร์วิสอินสแตนซ์กับสมาชิกภายใน ServiceGroup

ตารางที่ 7-3 แสดงอินเตอร์เฟสของกริดเซอร์วิส

ซึ่ง OGSI portType ทั้งหมดนี้กำหนดในขอบเขตชื่อของ OGSI

7.6.1 GridService portType

GridService portType จะเป็นส่วนที่ใช้สร้างกริดเซอร์วิสทั้งหมดและใช้จำกัดความพื้นฐานใน OGSI โดยลักษณะของการใช้ portType นั้นจะคล้ายคลึงกันกับการใช้อ็อบเจ็คคลาส เหมือนใน object-oriented programming language เช่น smalltalk หรือ Java ซึ่งจะซ่อนส่วนต่างๆเป็นองค์ประกอบย่อยๆ โดย GridService portType นั้นจะใช้ในการ คิวรี่และอัพเดต serviceData ของกริดเซอร์วิสอินสแตนซ์ และใช้จัดการการยกเลิกอินสแตนซ์

ในรูปแบบของเว็บเซอร์วิสนั้น จะเลือกวิธีอย่างใดอย่างหนึ่งระหว่าง ใช้รูปแบบการส่งเอกสาร หรือ ใช้ remote procedure call (RPC) แต่กริดเซอร์วิสนั้นได้ถูกออกแบบให้อิสระ สามารถใช้งานทั้ง 2 รูปแบบผสมผสานกันได้

7.6.2 HandleResolver PortType

HandleResolver portType นั้นเป็นส่วนที่ใช้ resolve ค่า GSH ให้เป็น GSR โดยเป็นอิสระกับ URI ใดๆของ GSH ซึ่งค่าเซอร์วิสอินสแตนซ์นั้นจะทำผ่าน HandleResolver portType ที่เรียกว่า handle resolver

แต่ละ handle resolver นั้นจะมีวิธีการที่ต่างกันเกี่ยวกับการแม็พจาก GSH ให้เป็น GSR โดยบาง handle resolver นั้นจะใช้เซอร์วิสของส่วนที่จัดการเกี่ยวกับไลฟ์ไทม์ของ hosting environment มาใช้โดยตรง เช่น การสร้างและการทำลายอินสแตนซ์แบบอัตโนมัติ หรือ การเพิ่มและลบการแม็พ เมื่อเซอร์วิสอินสแตนซ์นั้นลงทะเบียนว่าคงอยู่กับ resolver แล้ว resolver จะคิวรี่ค่า gridServiceHandle และค่า gridServiceReference ของอินสแตนซ์นั้นเพื่อแม็พลงฐานข้อมูล

portType นี้จะสืบเนื่องมาจาก GridService portType

7.6.3 Notification

จุดประสงค์ของ notification คือส่งข้อความที่น่าสนใจจาก notification source ไปที่ notification sink โดยจะอธิบายส่วนต่างๆไว้ตามนี้

- notification source คือ กริดเซอร์วิสอินสแตนซ์ที่ใช้ทำ NotificationSource portType และเป็นผู้ส่ง notification messages โดยที่แหล่งข้อมูลนั้นอาจจะส่ง notification message ไปที่ sink ใดๆก็ได้
- notification sink เป็นกริดเซอร์วิสอินสแตนซ์ที่ได้รับ notification message จากแหล่งข้อมูลใดๆ โดย sink จะใช้ทำ NotificationSink portType ซึ่งยอมให้รับค่า notification message
- notification message เป็นค่า XML ที่ส่งจาก notification source ไปยัง notification sink โดยค่า XML นี้จะถูกเลือกโดย subscription expression
- subscription expression เป็นค่า XML ที่ใช้อธิบายว่าข้อความนั้นควรจะส่งจาก notification source ไปยัง notification sink โดยขึ้นอยู่กับค่า serviceDataValues ของเซอร์วิสอินสแตนซ์
- ในการหาค่าว่า notification message อะไรจะต้องถูกส่งไปที่ไหนนั้น จะใช้การร้องขอค่า subscription ไปที่แหล่งข้อมูล ประกอบไปด้วย subscription expression, locator ของ notification sink ที่ notification message จะส่งไป และค่าเริ่มต้นของเวลาชีวิตสำหรับ subscription
- การร้องขอ subscription นั้นจะทำให้เกิดการสร้าง กริดเซอร์วิสอินสแตนซ์ ที่เรียกว่า subscription ที่ใช้สร้าง NotificationSubscription portType โดย portType นี้จะถูกใช้โดยไคล์เอนท์เพื่อจัดการไลฟ์ไทม์ของ subscription และค้นหาคุณสมบัติของค่า subscription

7.6.3.1 NotificationSource PortType

Notification portType จะยอมให้ client เพื่อลงท้าย notification message จากกริดเซอร์วิสอินสแตนซ์ที่ใช้ทำ portTypr นี้ โดยที่ NotificationSource portType จะขยายมาจาก GridService portType

7.6.3.2 NotificationSink portType

NotificationSink portType นั้นกำหนดโดยคำสั่งเดียวที่ใช้สำหรับส่ง notification message ไปให้เซอร์วิสอินสแตนซ์ที่ใช้ทำคำสั่งนั้น

7.6.4 Factory PortType

factory เป็นรูปแบบที่ตรงตามกริดเซอร์วิสที่ไคล์เอนท์สร้างขึ้นโดยกริดเซอร์วิสอินสแตนซ์อื่นๆ โดยที่ไคล์เอนท์จะเรียกสร้างคำสั่งบน factory และได้รับผลตอบรับเป็น locator ของเซอร์วิทอินสแตนซ์ใหม่ที่ถูกสร้างขึ้น โดย factory อาจเป็นกริดเซอร์วิสอินสแตนซ์ที่ใช้สร้าง Factory portType หรืออาจเป็นกริดเซอร์วิสอินสแตนซ์ที่สร้างคำสั่งของ factory โดยเฉพาะ เช่น NotificationSource::subscribe เป็นคำสั่งที่อธิบายไปในเรื่องที่แล้ว

บนการสร้าง factory นั้น กริดเซอร์วิสอินสแตนซ์ที่ถูกสร้างขึ้นใหม่นั้นจะถูกลงทะเบียนและได้รับค่า GSH ซึ่งเป็น handle resolution service ที่ใช้ในการลงทะเบียนกับ hosting environment

ซึ่ง Factory portType จะต้องสืบเนื่องมาจาก GridService portType

7.6.5 ServiceGroup portType

ServiceGroup เป็นกริดเซอร์วิสอินสแตนซ์ที่ดูแลข้อมูลข่าวสารเกี่ยวกับกลุ่มของกริดเซอร์วิสอื่นๆ เช่น กริดเซอร์วิสที่ใช้ลงทะเบียนจะกำหนดโดย portType ที่สืบเนื่องมาจากพฤติกรรมพื้นฐานจาก ServiceGroup ซึ่งมี 3 portType ที่ดูแลรูปแบบของกลุ่มของเซอรีวิสนี้ คือ ServiceGroup, ServiceGroupEntry และ ServiceGroupRegistration

ServiceGroup portType เป็นส่วนที่ใช้จัดเตรียมรูปแบบการแทน service group โดยมีอย่างน้อย 0 เซอร์วิสที่เป็นสมาชิกในกลุ่ม โดยค่า SDE ของ ServiceGroup นั้นจะประกอบไปด้วยค่าสำหรับแต่ละกริดเซอร์วิสที่เป็นสมาชิกในกลุ่ม โดยมีค่า ServiceGroupEntryLocator ใช้อ้างที่เซอร์วิสอินสแตนซ์ที่ใช้ทำ ServiceGroupEntry portType ซึ่งเตรียมฟังก์ชั่นควบคุมการดูแลไว้ โดยจะมีค่าที่เป็นเอกเทศไว้ใช้เป็นคีย์คือ GSH สำหรับแต่ละค่า

คุณสมบัติต่อไปนี้จะเป็นคุณสมบัติที่จำเป็นของ ServiceGroup, ServiceGroupEntry, ServiceGroupRegistration และ สมาชิกของกริดเซอร์วิสอินสแตนซ์ของ ServiceGroup

- กริดเซอร์วิสอินสแตนซ์จะต้องมีสมาชิกของหลายๆ ServiceGroups
- สมาชิกของกริดเซอร์วิสอินสแตนซ์ของ ServiceGroup จะต้องทำในหลายๆ portType ที่ต่างกัน
- ค่า ServiceGroupEntry จะต้องถอดออกจาก ServiceGroup โดยการจัดการไลฟ์ไทม์ของ ServiceGroupEntry
- เมื่อ ServiceGroup ถูกทำลาย client จะต้องไม่ได้รับผลกระทบจากการทำลายเซอร์วิส ServiceGroupEntry นั้น
- เมื่อ ServiceGroup ถูกทำลาย client จะต้องสามารถทราบเกี่ยวกับการไม่อยู่ของ ServiceGroupEntry หรือรายละเอียดที่ถูกต้องของมัน (เช่น คุณสมบัติของไลฟ์ไทม์)
- ServiceGroupEntry จะต้องเป็นของหนึ่ง ServiceGroup

- ถ้า GridService รวมค่าการยกเลิก ServiceGroup ไว้ด้วยแล้ว ServiceGroup จะต้องไม่สนใจมัน
- สมาชิกทั้งหมดของกริดเซอร์วิสอินสแตนซ์ของ ServiceGroup จะต้องทำตามกันอย่างน้อย 1 portType ที่เป็นสมาชิกใน membershipContentRule ของ ServiceGroup
- กริดเซอร์วิสอินสแตนซ์ที่เป็นสมาชิกจะต้องรวมอยู่ใน ServiceGroup โดยที่ ServiceGroup เปรียบเสมือนถุงที่รวบรวมค่าต่างๆไว้ โดยผู้ออกแบบได้ใช้สืบเนื่องมาจาก ServiceGroup ซึ่งเป็นเซ็ทของกลุ่มต่างๆ

บทที่ 8
การเขียนกริดเซอร์วิส

ในส่วนนี้จะแสดงขั้นตอนการเขียนกริดเซอร์วิสซึ่งได้แบ่งเป็น 5 ขั้นตอน โดยในตัวอย่างนี้จะเป็น กริดเซอร์วิสทางคณิตศาสตร์อย่างง่าย คือ จะมีคำสั่งบวกและลบ

1. กำหนดรูปแบบของเซอร์วิส ส่วนนี้จะทำโดย GWSDL
2. สร้างเซอร์วิส ส่วนนี้ทำโดยใช้ภาษา Java
3. กำหนดค่าส่วนที่จะนำมาใช้งาน ส่วนนี้จะทำโดย WSDD
4. สร้างไฟล์ชนิด GAR จากการคอมไพล์ทุกส่วนแล้วนำมารวมกัน ส่วนนี้จะใช้ Ant ช่วย
5. นำเซอร์วิสไปไว้ในกริดเซอร์วิสคอนเทนเนอร์ ส่วนนี้จะใช้ Ant ช่วย

8.1 ขั้นตอนที่ 1: กำหนดรูปแบบของเซอร์วิส

ขั้นตอนแรกจะเป็นการเขียนกริดเซอร์วิส หรือ เว็บเซอร์วิส ที่ใช้กำหนดรูปแบบของเซอร์วิสนั้น โดยจะเป็นการกำหนดว่าจะให้เซอร์วิสนี้มีความสามารถอะไรบ้างที่จะให้ผู้อื่นใช้งาน โดยที่ส่วนนี้จะไม่สนใจว่าจะใช้อัลกอริธึมยังไงในการเขียน จะต้องใช้งานไฟล์ไหนบ้าง แต่จะสนใจเพียงว่ามีคำสั่งอะไรบ้างให้ใช้งาน โดยรูปแบบการใช้งานในกริดเซอร์วิสหรือเว็บเซอร์วิสจะใช้งานผ่านอินเตอร์เฟสที่เรียกว่า portType ซึ่งส่วนนี้จะกำหนดโดย Web Service Description Language (WSDL)

วิธีการที่จะกำหนด WSDL ที่จะให้ใช้งานในส่วนนี้มีวิธีทำได้อยู่ 2 วิธี ดังต่อไปนี้

- เขียน WSDL ด้วยตนเอง จะสามารถกำหนดทุกอย่างได้ตามที่ตนเองต้องการ แต่ผู้ใช้มักจะไม่คุ้นเคยกับส่วนนี้ เนื่องจาก WSDL เป็นภาษาที่เขียนหลายส่วนมากเกินความจำเป็น
- สร้าง WSDL ขึ้นมาตามรูปแบบของ Java ซึ่งจะสร้างออกมาได้สะดวก แต่อาจไม่ครอบคลุมทุกอย่าง มักมีปัญหาในการทำงานบางอย่าง

โดยตัวอย่างการกำหนดส่วนนี้ทำได้ดังโค้ดตัวอย่างด้านล่างนี้

```
public interface Math
{
  public void add(int a);
  public void subtract(int a);
  public int getValue();
}
```

คราวนี้จะทำให้อยู่ในรูปแบบที่อธิบายด้วย WSDL ที่แม้ว่าจะเข้าใจยากกว่ารูปแบบของ Java แต่การทำงานของมันจะเหมาะสมกับกริดมากกว่า ซึ่งกริดนั้นจะเปลี่ยนแปลงบางส่วนของ WSDL ให้

เป็น GWSDL ซึ่งกำหนดรูปแบบมาจาก OGSI และจาก Globus Toolkit โดย GWSDL นั้นเป็นการขยายส่วนของ WSDL ให้รองรับการอธิบายเกี่ยวกับกริดเซอร์วิส ที่ไม่สามารถอธิบายได้ด้วย WSDL โดยรูปแบบของ GSWDL ที่เหมือนกับที่ได้กำหนดในรูปแบบของ Java ข้างต้นนั้น แสดงได้ดังโค้ดด้านล่างนี้

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
name="MathService"

targetNamespace="http://www.globus.org/namespaces/2004/02/pr
ogtutorial/MathService"

xmlns:tns="http://www.globus.org/namespaces/2004/02/progtuto
rial/MathService"

xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI
"

xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gri
dWSDLExtensions"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

<import location="../../ogsi/ogsi.gwsdl"

namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"
/>

<types>
<xsd:schema
targetNamespace="http://www.globus.org/namespaces/2004/02/pr
ogtutorial/MathService"
  attributeFormDefault="qualified"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="add">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="value" type="xsd:int"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="addResponse">
    <xsd:complexType/>
  </xsd:element>
  <xsd:element name="subtract">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="value" type="xsd:int"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="subtractResponse">
    <xsd:complexType/>
```

```
    </xsd:element>
    <xsd:element name="getValue">
      <xsd:complexType/>
    </xsd:element>
    <xsd:element name="getValueResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="value" type="xsd:int"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  </types>

  <message name="AddInputMessage">
    <part name="parameters" element="tns:add"/>
  </message>
  <message name="AddOutputMessage">
    <part name="parameters" element="tns:addResponse"/>
  </message>
  <message name="SubtractInputMessage">
    <part name="parameters" element="tns:subtract"/>
  </message>
  <message name="SubtractOutputMessage">
    <part name="parameters" element="tns:subtractResponse"/>
  </message>
  <message name="GetValueInputMessage">
    <part name="parameters" element="tns:getValue"/>
  </message>
  <message name="GetValueOutputMessage">
    <part name="parameters" element="tns:getValueResponse"/>
  </message>

  <gwsdl:portType name="MathPortType"
  extends="ogsi:GridService">
    <operation name="add">
      <input message="tns:AddInputMessage"/>
      <output message="tns:AddOutputMessage"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
    <operation name="subtract">
      <input message="tns:SubtractInputMessage"/>
      <output message="tns:SubtractOutputMessage"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
    <operation name="getValue">
      <input message="tns:GetValueInputMessage"/>
      <output message="tns:GetValueOutputMessage"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
  </gwsdl:portType>
  </definitions>
```

หมายเหตุ: โค้ดส่วนที่ทำเน้นตัวอักษรไว้ เป็นส่วนที่สำคัญที่จะนำมาอธิบายเพิ่มเติม ดังต่อไปนี้

ส่วนแรกเป็นเป้าหมายของเนมสเปซจะเป็นอย่างไร

```
http://www.globus.org/namespaces/2004/02/progtutorial/MathSe
rvice
```

ส่วนที่ 2 เป็นการประกาศเนมสเปซของ OGSI

```
xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI
"
xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gri
dWSDLExtensions"
```

ส่วนที่ 3 เป็นการอิมพอร์ท GWSDL และกำหนดชนิดข้อความ และ portType ของ OGSI

```
<import location="../../ogsi/ogsi.gwsdl"
namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"
/>
```

8.1.1 ส่วนที่แตกต่างกันระหว่าง GWSDL และ WSDL

เมื่อพิจารณาจากโค้ดนี้จะแสดงให้เห็นว่าส่วนที่ขยายขึ้นมาจาก WSDL นั้น คืออะไรบ้าง ดังนี้

```
<gwsdl:portType name="MathPortType"
extends="ogsi:GridService"> </gwsdl:portType>
```

ส่วนแรกที่สำคัญคือ แท็กของ GWSDL ที่แสดงด้วย namespace: <gwsdl:portType> ส่วนนี้จะทำให้สามารถกำหนด portType ให้ขยายได้จาก 1 portType ที่ขยายมาจากหลายๆ portType

ส่วนที่สองที่สำคัญคือ ความเกี่ยวข้องกับ ข้อมูลเซอร์วิส ซึ่งจะแสดงให้เห็นในเรื่องต่อๆไป

8.2 ขั้นตอนที่ 2: สร้างเซอร์วิส

จากขั้นตอนแรกที่กล่าวไปเป็นการบอกว่า เซอร์วิสจะทำอะไร ส่วนต่อไปนี้จะเป็นการแสดงว่าจะทำอย่างที่กำหนดไว้ในขั้นตอนแรกอย่างไร กล่าวคือ เขียนโปรแกรมให้ทำตามที่กำหนดไว้นั่นเอง ซึ่งใช้ Java ในการเขียนโค้ดในส่วนนี้ โดยจะอธิบายแยกเป็นส่วนๆดังต่อไปนี้

ส่วนแรก เป็นการอิมพอร์ทแพ็คเกจที่ต้องการใช้งาน

```
package org.globus.progtutorial.services.core.first.impl;

import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import
org.globus.progtutorial.stubs.MathService.MathPortType;
import java.rmi.RemoteException;
```

ส่วนนี้จะเป็นการกำหนดคลาสที่ชื่อว่า MathImpl ที่จะใช้สร้างเซอร์วิสที่กำหนดไว้

```
public class MathImpl extends GridServiceImpl implements
MathPortType
```

ส่วนนี้จะเป็นการสร้าง คอนสตัคเตอร์ ของกริดเซอร์วิส

```
public MathImpl()
{
        super("Simple Math Service");
}
```

ส่วนนี้จะเป็นการกำหนดวิธีการที่จะให้ได้ผลลัพธ์ดังที่กำหนดไว้จากขั้นตอนแรก

```
public void add(int a) throws RemoteException
```

```
{
  value = value + a;
}

public void subtract(int a) throws RemoteException
{
  value = value - a;
}

public int getValue() throws RemoteException
{
  return value;
}
```

เมื่อนำโค้ดแต่ละส่วนมารวมกันแล้ว จะได้โค้ดดังต่อไปนี้

```
package org.globus.progtutorial.services.core.first.impl;

import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import
org.globus.progtutorial.stubs.MathService.MathPortType;
import java.rmi.RemoteException;

public class MathImpl extends GridServiceImpl implements
MathPortType
{
  private int value = 0;

  public MathImpl()
  {
    super("Simple MathService");
  }

  public void add(int a) throws RemoteException
  {
    value = value + a;
  }

  public void subtract(int a) throws RemoteException
  {
    value = value - a;
  }

  public int getValue() throws RemoteException
  {
    return value;
  }
}
```

8.3 ขั้นตอนที่ 3: กำหนดค่าส่วนที่จะนำมาใช้งาน

เมื่อมาถึงขั้นตอนนี้ เราได้ทำส่วนที่สำคัญก็คือ กำหนดรูปแบบของเซอร์วิส และ สร้างเซอร์วิส

มาแล้ว ในขั้นตอนนี้จะนำ 2 ส่วนแรกมารวมเข้าด้วยกัน และทำให้มันเป็นกริดเซอร์วิส ที่จะใช้งานบน Server 1 ได้ ในขั้นนี้มักจะเรียกว่าการ ดีพลอย กริดเซอร์วิส (deployment grid service)

ส่วนที่สำคัญในขั้นตอนนี้คือการกำหนดรายละเอียดการดีพลอยที่เรียกว่า deployment descriptor ซึ่งเป็นไฟล์ที่จะบอก server ว่าจะสร้างกริดเซอร์วิสขึ้นมาอย่างไร (เช่น บอกว่า GSH ของเราจะเป็นอะไร) โดยค่ารายละเอียดการดีพลอยนั้น จะเขียนในรูปแบบของ WSDD (Web Service Deployment Descriptor) ซึ่งมีตัวอย่างโค้ดให้ดู ดังนี้

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig"
xmlns="http://xml.apache.org/axis/wsdd/"

xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

  <service name="progtutorial/core/first/MathService"
provider="Handler" style="wrapped">
    <parameter name="name" value="MathService"/>
    <parameter name="className"
value="org.globus.progtutorial.stubs.MathService.MathPortTyp
e"/>

    <parameter name="baseClassName"
value="org.globus.progtutorial.services.core.first.impl.Math
Impl"/>
    <parameter name="schemaPath"
value="schema/progtutorial/MathService/Math_service.wsdl"/>

    <!-- Start common parameters -->
    <parameter name="allowedMethods" value="*"/>
    <parameter name="persistent" value="true"/>
    <parameter name="handlerClass"
value="org.globus.ogsa.handlers.RPCURIProvider"/>
  </service>
</deployment>
```

จะอธิบายส่วนสำคัญแต่ละส่วน ต่อไปนี้

8.3.1 ชื่อเซอร์วิส

คือโค้ดส่วนนี้

```
  <service name="progtutorial/core/first/MathService"
provider="Handler" style="wrapped">
```

เป็นส่วนที่กำหนดว่า กริดเซอร์วิสของเราจะเจอที่ตำแหน่งไหน เราอาจจะนำส่วนนี้มารวมกับที่อยู่ของกริดเซอร์วิสคอนเทนเนอร์ก็ได้ โดยเราจะได้ค่า GSH แบบเต็มจากกริดเซอร์วิส เช่น ถ้าเราใช้ GT3 ที่มีคอนเทนเนอร์เป็นแบบสแตนด์อโลนจะทำให้มีที่อยู่ที่เป็นฐานคือ http://localhost:8080/ogsa/services เมื่อรวมกับใน GSH ของเซอร์วิสที่เรากำหนดไว้แล้ว จะเป็นดังนี้ http://localhost:8080/ogsa/services/progtutorial/core/first/MathService

8.3.2 ชื่อเซอร์วิสอีกส่วน

คือโค้ดส่วนนี้

```
<parameter name="name" value="MathService"/>
```

ส่วนนี้จะเป็นส่วนที่บ่งบอกว่าเซอร์วิสนั้นจะมีค่าทั้งเป็น ชื่อแอททิบิวท์ อย่างที่แสดงไว้ส่วนแรก และส่วนนี้จะบ่งบอกถึงพารามิเตอร์ของเซอร์วิส ในโค้ดจะใช้เป็น MathService

8.3.3 className และ baseClassName

คือโค้ดส่วนนี้

```
<parameter name="className"
value="org.globus.progtutorial.stubs.MathService.MathPortTyp
e"/>
<parameter name="baseClassName"
value="org.globus.progtutorial.services.core.first.impl.Math
Impl"/>
```

ใน WSDD นั้น ค่านี้จะบอกว่าคลาสใดที่ใช้สร้างเซอร์วิส (ในตัวอย่างนี้จะใช้ MathImpl จากส่วนก่อนๆ) อย่างไรก็ตามควรมีการแจ้งค่านี้ว่าเป็น Math-PortType stub ที่กำหนดต่อเนื่องมาจากหน้าก่อนๆ

เมื่อกริดเซอร์วิสได้กำหนดให้ยืดหยุ่นมากกว่าเว็บเซอร์วิส จึงจำเป็นต้องกำหนดความแตกต่างกันของ className และ baseClassName โดย className จะอ้างรูปแบบที่แสดงฟังก์ชั่นของกริดเซอร์วิส (MathPortType) และ baseClassName จะเป็นคลาสที่จัดเตรียมการสร้างกริดเซอร์วิส ในกรณีนี้ baseClassName จะเป็น MathImpl ที่สร้างมาจากส่วนก่อนๆ ในขั้นตอนต่อไป จะแสดงให้เห็นว่า baseClassName ไม่ได้ต้องการรวมส่วนที่สร้างขึ้นมาทั้งหมดของสิ่งที่แสดงใน className

8.3.4 ไฟล์ WSDL

คือโค้ดส่วนนี้

```
<parameter name="schemaPath"
value="schema/progtutorial/MathService/Math_service.wsdl"/>
```

เป็น schemaPath ที่บอกกริดเซอร์วิสคอนเทนเนอร์ว่า WSDL ไฟล์นั้นจะสามารถเจอได้ที่ไหน ที่ไม่ได้บอกถึง GWSDL เนื่องจาก GWSDL ไม่ใช่มาตรฐาน แต่เป็นส่วนที่ขยายมาจาก WSDL โดยขั้นแรกจะต้องแปลงให้เป็น WSDL เพื่อใช้บอกว่ามีการใช้งานกับเทคโนโลยีเว็บเซอร์วิสที่มีอยู่จริง ซึ่งไฟล์ WSDL นั้นจะสร้างโดยอัตโนมัติด้วย GT3 เมื่อเราคอมไพล์เซอร์วิส

8.3.5 พารามิเตอร์ทั่วไป

คือโค้ดส่วนนี้ ซึ่งเป็นค่าพารามิเตอร์ที่ใช้ในโปรแกรมตัวอย่าง

```
<!-- Start common parameters -->
<parameter name="allowedMethods" value="*"/>
<parameter name="persistent" value="true"/>
<parameter name="handlerClass"
value="org.globus.ogsa.handlers.RPCURIProvider"/>
```

8.4 ขั้นตอนที่ 4: สร้างไฟล์ชนิด GAR จากการคอมไพล์ทุกส่วนแล้วนำมารวมกัน

เมื่อมาถึงขั้นตอนนี้เราจะได้ รูปแบบของเซอร์วิสจากขั้นที่ 1 ได้เซอร์วิสที่สร้างขึ้นจากขั้นที่ 2 และได้ส่วนที่บอกกริดเซอร์วิสคอนเทนเนอร์ว่าจะแสดงส่วนที่ได้จากขั้นที่ 1 และ 2 ไปให้ภายนอกเห็นอย่างไรบ้าง แต่ส่วนที่ได้มานั้นยังไม่สามารถใช้งานได้จริงในคอนเทนเนอร์ เนื่องจากไฟล์ Java ก็ยังไม่ได้คอมไพล์และไฟล์อื่นๆก็ยังไม่รู้ว่าต้องนำไปวางที่ส่วนไหน

ดังนั้นในขั้นนี้เราจะรวมทุกส่วนที่กล่าวมาเข้าด้วยกัน เพื่อให้นำไปใช้งานจริงได้ โดยเราจะสร้างไฟล์ Grid Archive หรือไฟล์ประเภท GAR ขึ้นมา โดยที่ไฟล์ Gar นั้นจะเป็นไฟล์เดี่ยวที่บรรจุไฟล์ทั้งหมดและข้อมูลที่กริดเซอร์วิสคอนเทนเนอร์ต้องการทั้งหมดไว้เพื่อที่จะดีพลอยไปให้แก่เซอร์วิสที่เราสร้างขึ้นให้ข้างนอกมาใช้งาน ต่อมาเราจะแสดงวิธีที่จะทำไฟล์ GAR และดีพลอยมัน โดยขั้นตอนนั้นจะแบ่งออกได้ดังนี้

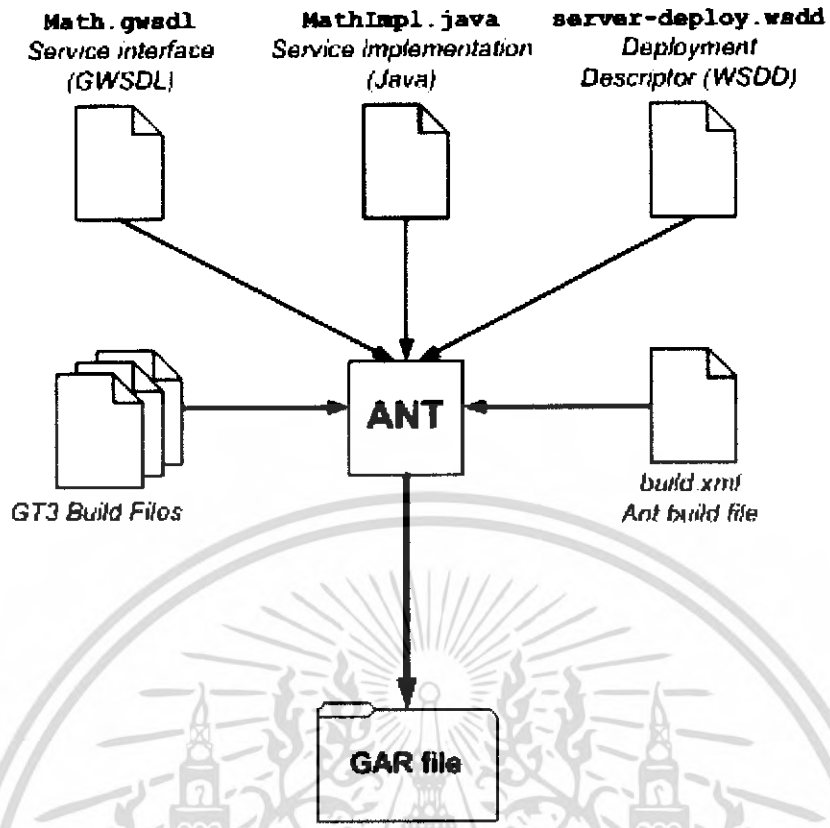
- แปลงจาก GWSDL ให้เข้าสู่รูปแบบมาตรฐานคือ WSDL
- สร้างสตับคลาสจาก WSDL
- คอมไพล์สตับคลาสนั้นๆ
- คอมไพล์เซอร์วิสที่เราสร้างขึ้น
- รวบรวมไฟล์ทั้งหมดเข้าสู่โครงสร้างที่จะใช้โดยเฉพาะ

ขั้นตอนเหล่านี้ เมื่อกล่าวขึ้นมาอาจจะฟังดูยาก แต่ Globus ก็ได้มีเครื่องมือที่ช่วยในขั้นตอนนี้นั่นก็คือ Ant ที่สามารถทำขั้นตอนที่กล่าวมาได้ในขั้นตอนเดียว

8.4.1 Ant

Ant เป็น ซอฟท์แวร์ของ Apache ที่เป็นเครื่องมือสร้าง Java โดยอาศัยหลักการคล้ายคำสั่ง make ของ UNIX ซึ่งทำให้คนเขียนโปรแกรมสามารถข้ามขั้นตอนที่ยุ่งยากเกี่ยวกับการนำไฟล์เอ็กคิวท์จากไฟล์ต้นทาง ซึ่งส่วนนี้จะทำโดย Ant ที่จะช่วยลดขั้นตอนในส่วนนี้ให้เหลือขั้นเดียว ซึ่งด้วย Ant นี้จะทำให้ปัญหาของการทำกริดเซอร์วิสนั้นมาอยู่ที่การกำหนดรูปแบบและการสร้างมันขึ้นมา

ซึ่ง Ant ต้องการไฟล์ที่ใช้สร้างอยู่ 2 ไฟล์ คือ ไฟล์ที่ใช้สร้างที่เป็นส่วนหนึ่งของ GT3 และอีกส่วนคือจะเป็นส่วนที่สำคัญที่เราจะต้องเขียนขึ้นมาเอง (ตามขั้นตอนที่สร้างโค้ด WSDL และสร้างสตับ) เมื่อสร้างไฟล์นี้ขึ้นมาแล้วอาจจะใช้ได้กับหลายๆกริดเซอร์วิสโดยไม่ต้องสร้างขึ้นมาใหม่ ซึ่งส่วนนี้ Ant ก็ได้ทำไว้ให้แล้ว แต่ถ้าต้องการงานที่ยืดหยุ่นมากขึ้นก็คงต้องสร้างไฟล์นี้ขึ้นมาใหม่ โดยเราจะแสดงองค์ประกอบของส่วนนี้ให้เห็นดังรูปต่อไปนี้



รูปที่ 8-1 แสดงองค์ประกอบในการใช้งาน Ant

สิ่งที่จำเป็นต้องทำในขั้นตอนนี้

จำเป็นต้องใส่ไดเรคทอรี่ที่เราลง GT3 ไว้ ดังในตัวอย่างข้างล่างนี้

```
ogsa.root=/usr/local/gt3
```

สร้างเซอร์วิสที่เป็น GAR (ตัวอย่างนี้จะสร้าง MathService) โดยใส่ สคริปท์ ตามนี้

```
./tutorial_build.sh <service base directory> <service's
GWSDL file>
```

โดยที่ <service base directory> นั้นเป็นไดเรคทอรี่ที่เราวาง server-deploy.wsdd ไว้และกำหนดที่ที่จะเจอ MathImpl.java ดังตัวอย่างข้างล่างนี้

```
./tutorial_build.sh\org/globus/progtutorial/services/core/
first
\schema/progtutorial/MathService/Math.gwsdl
```

ถ้าทุกขั้นตอนนั้นถูกต้องไฟล์ GAR นั้นจะอยู่ในตำแหน่งของ $TUTORIAL_DIR/build/lib. ดังในตัวอย่างนี้

```
$TUTORIAL_DIR/build/lib/org_globus_progtutorial_services_cor
e_first.gar
```

8.5 ขั้นตอนที่ 5: นำเซอร์วิสไปไว้ในกริดเซอร์วิสคอนเทนเนอร์

ไฟล์ GAR จากขั้นตอนที่แล้วนั้นจะบรรจุไฟล์ที่จำเป็นที่จะใช้ในกริดเซอร์วิสแล้ว ขั้นนี้เราดี

พลอยมันโดย Ant ซึ่งจะนำไฟล์เหล่านี้ คือ WSDL, สตับที่คอมไพล์แล้ว, โค้ดที่สร้างขึ้นคอมไพล์แล้ว และก็ WSDD ไปไว้ที่ตำแหน่งที่ให้ใช้งานที่กำหนดไว้ใน GT3 ซึ่งคำสั่งเกี่ยวกับการดีพลอยนั้นเราจะต้องทำที่ root ของส่วนที่เราลง GT3 โดยใช้คำสั่งตามรูปแบบนี้

```
ant deploy -Dgar.name=<full path of GAR file>
```

สำหรับในตัวอย่างนี้จะได้โค้ดดังนี้

```
ant deploy \ -Dgar.name=$TUTORIAL_DIR/build/lib/
org_globus_progtutorial_services_core_first.gar
```

การดีพลอยในขั้นตอนสุดท้ายนี้ก็ทำแค่เพียงเท่านี้ ซึ่งเมื่อมาถึงขั้นนี้เราจะได้ 5 ขั้นตอนที่จำเป็นของการทำกริดเซอร์วิสแล้ว ต่อมาเราจะมาแสดงขั้นตอนในการเขียนแอพพลิเคชั่นในฝั่ง client

8.6 แอพพลิเคชั่นฝั่งไคล์เอนท์

เมื่อเราต้องการทดสอบกริดเซอร์วิสที่เขียนขึ้นด้วยคอมมานด์-ไลน์ ซึ่งจะใช้วิธี getValue ในการทดสอบ MathService นี้ สิ่งที่ client จะต้องได้รับจากคอมมานด์-ไลน์มี 2 ค่าคือ

- GSH
- ค่าที่เพิ่มขึ้นตามการทำงานที่กำหนดไว้ในเซอร์วิส

เราใช้โค้ดนี้ในการทดสอบการทำงานของ MathService

```
package org.globus.progtutorial.clients.MathService;

import
org.globus.progtutorial.stubs.MathService.service.MathServic
eGridLocator;
import
org.globus.progtutorial.stubs.MathService.MathPortType;

import java.net.URL;

public class Client
{
  public static void main(String[] args)
  {
    try
    {
      // Get command-line arguments
      URL GSH = new java.net.URL(args[0]);
      int a = Integer.parseInt(args[1]);


      // Get a reference to the MathService instance
      MathServiceGridLocator mathServiceLocator = new
MathServiceGridLocator();
```

```
            MathPortType math =
        mathServiceLocator.getMathServicePort(GSH);

            // Call remote method 'add'
            math.add(a);
            System.out.println("Added " + a);

            // Get current value through remote method 'getValue'
            int value = math.getValue();
            System.out.println("Current value: " + value);
          }catch(Exception e)
          {
            System.out.println("ERROR!");
            e.printStackTrace();
          }
        }
      }
```

แต่ก่อนที่เราจะรันคอมไพเลอร์นี้ สิ่งที่ต้องรันก่อน มีดังต่อไปนี้

```
source $GLOBUS_LOCATION/etc/globus-devel-env.sh
```

เพื่อที่จะคอมไพล์ client จะต้องทำตามนี้

```
javac \ -classpath ./build/classes/:$CLASSPATH \
org/globus/progtutorial/clients/MathService/Client.java
```

โดยก่อนที่จะให้มันทำงานเราต้องเริ่มการทำงานของคอนเทนเนอร์ก่อน โดยรันคำสั่งนี้ที่ root

```
globus-start-container
```

เมื่อคอนเทนเนอร์ทำงานแล้ว เราจะเห็นรายการของ GSH ที่มีขึ้นมา ซึ่งเราต้องเลือก MathService ตามนี้

```
http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/
MathService
```

คราวนี้จะเป็นการสรุปคำสั่งเต็มๆที่จะให้ client ทำงาน

```
java \
-classpath ./build/classes/:$CLASSPATH \
org.globus.progtutorial.clients.MathService.Client \
http://127.0.0.1:8080/ogsa/services/progtutorial/core/first/
MathService \
5
```

ถ้าคำสั่งนี้ผ่านเราจะเห็นค่าต่อไปนี้

Added 5

Current value: 5

ซึ่งแสดงว่าคำสั่งบวกของเราใช้งานได้แล้ว โดยจะเริ่มบวกค่าจากค่าเริ่มต้นที่เป็น 0 ทำให้ได้ค่า 5 ขึ้นมา เมื่อเรารันคำสั่งนั้นอีกครั้ง เราจะได้ค่าดังต่อไปนี้

Added 5

Current value: 10

ซึ่งแสดงว่าสเตทภายในของมันนั้นยังเก็บค่า 5 ซึ่งเป็นค่าเดิมไว้อยู่ ทำให้จะได้ค่าใหม่ออกมาเป็น 10 ซึ่งเพิ่มจากค่าเดิมนั่นเอง

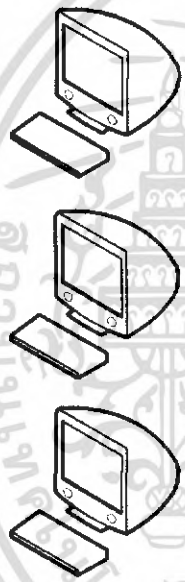
บทที่ 9

การทดลอง

การทดลองจะเป็นการทดสอบการใช้งานได้ของระบบกริดที่สร้างขึ้นและทดสอบการใช้งานได้ของโปรแกรมที่เขียนขึ้นมาโดยจะเน้นในเรื่องของประสิทธิภาพการเรียกใช้งานเซอร์วิสในรูปแบบต่างๆ เช่นการทดลองรันบนเครื่องเดียว หรือหลายๆเครื่อง แล้วสังเกตว่าผลที่ได้มีความแตกต่างกันอย่างไร และทดลองเรียกใช้เซอร์วิสจากเครื่องที่มีความเร็วต่างกันว่าให้ผลเป็นอย่างไร

9.1 อุปกรณ์ที่ใช้ในการทดลอง

ในการทดลอง จะใช้ คอมพิวเตอร์ 3 เครื่องในการสร้างระบบโดยมีสเปคที่ต่างกันดังนี้



Test1.kmitl.ac.th <161.246.6.112> RAM 256 M

Test2.kmitl.ac.th<161.246.6.113> RAM 192M

Test3.kmitl.ac.th <161.246.6.119> RAM 192 M

ตารางที่ 9-1 อุปกรณ์ที่ใช้ในการทดลอง

และใช้ VMWarc ช่วยในการทดลองเพื่อจำลอง OS linux

9.2 วิธีการทดสอบเพื่อเริ่มต้นการทดลอง

ทดสอบการใช้กริดโดยการเรียกใช้งาน test service ซึ่งมีอยู่แล้วใน gt3.2.1 โดยให้ทุกเครื่องลองเรียกใช้เซอร์วิสของกันและกัน

เช่น ให้ test1 ลองเรียกใช้เซอร์วิสของ test2

```
S managed-job-globusrun –factory \
    http://test2:8080/ogsa/services/base/gram/MasterForkManagedJobFactoryService -file \
```

$GLOBUS_LOCATION/schema/base/gram/example/test.xml

```
========== Status Notification ==========
Job Status: StageIn
==========================================
========== Status Notification ==========
Job Status: active
==========================================
========== Status Notification ==========
Job Status: done
==========================================
DESTROYING SERVICE
```

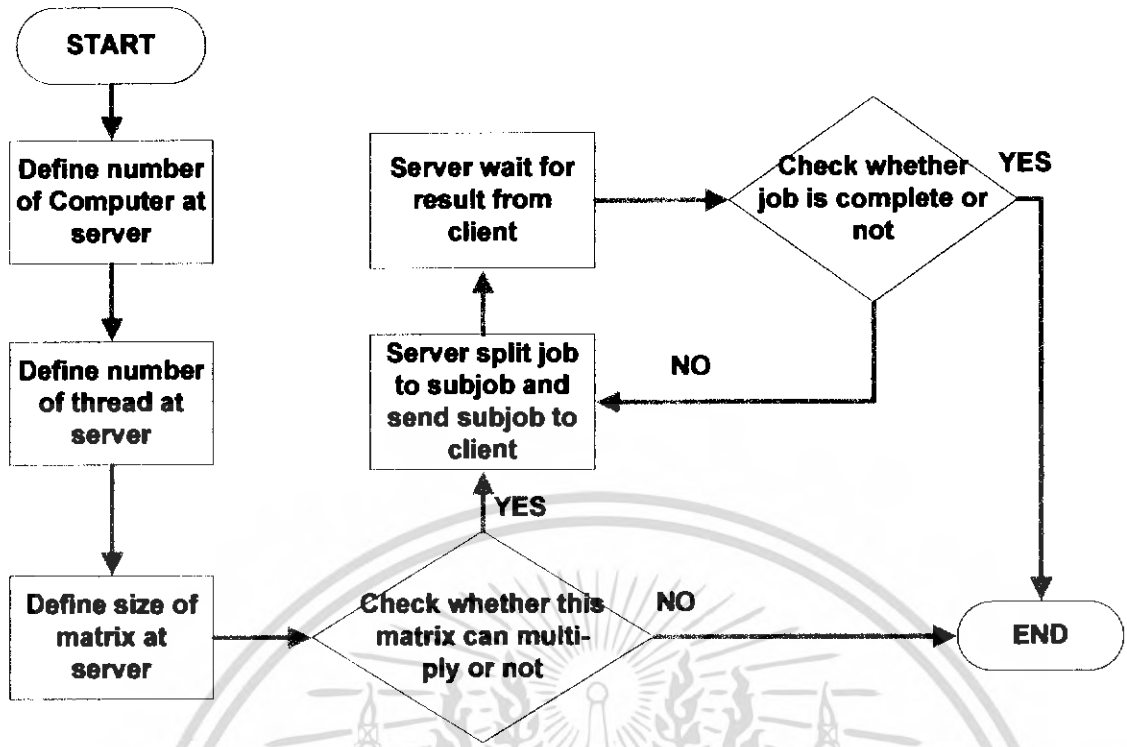
รูปที่ 9-2 ผลลัพธ์จากการรัน test service

ปรากฏว่าทุกเครื่องสามารถรันแล้วได้ผลดังรูป สามารถสรุปได้ว่าทุกเครื่องสามารถใช้งานระบบกริดที่สร้างขึ้นได้

9.3 ขั้นตอนในการทำการทดลอง

9.3.1 การทำงานของโปรแกรม

การทำงานของโปรแกรมดังที่ได้แสดงใน flowchart รูป 9-3 จะเป็นการทำงานของโปรแกรมที่เขียนขึ้นเพื่อกระจายงานที่มีอยู่ไปทำงานบนเครื่องอื่นๆ



รูปที่ 9-3 แสดงขั้นตอนในการทำงานของโปรแกรม

9.3.2 คำสั่งที่ใช้รันโปรแกรม

```
[third@test3 client]$ java GenerateMatrix aaa.txt 40 40
[third@test3 client]$ java GenerateMatrix bbb.txt 40 40
[third@test3 client]$ java src/Client aaa.txt bbb.txt ccc.txt 100
```

รูปที่ 9-4 แสดงคำสั่งที่ใช้รันโปรแกรม

- การคอมไพล์ ไฟล์ Client.java,Subjob.java,Data.java เข้าไปที่ไดเรกเทอรี่ $matrix/client
- การรันโปรแกรมต้องใช้คำสั่ง ดังรูปที่ 9-4 คือมีการสร้างเมตริกขนาดที่ต้องการ และใช้คำสั่งรันโปรแกรมจาวา โดยมีพารามิเตอร์ดังนี้ คือเทกซ์ไฟล์ 3 ไฟล์ คือ 2ไฟล์ที่เป็นตัวคูณ และไฟล์ที่3 เป็นผลลัพธ์ และพารามิเตอร์ตัวที่ 4 คือจำนวนthread ที่เราต้องการใช้ในการทำงาน

9.3.3 การกำหนดตัวแปรที่มีผลกับการทดลอง

ตัวแปรที่มีผลกับการทดลองมี 3 ตัวแปรด้วยกัน คือ

- ขนาดของเมตริก
- ขนาดของthread
- จำนวนของเครื่องที่ทำงาน
- ตัวแปรอื่นๆ (ที่ไม่สามารถวัดค่าที่แน่นอนได้ เช่น ความคับคั่งของเครือข่าย)

9.3.4 วิธีการทำการทดลอง

Thread / Matrix size	10			25			50			75			100		
	Number of machine			Number of machine			Number of machine			Number of machine			Number of machine		
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
20*20															
40*40															
80*80															
160*160															
320*320															

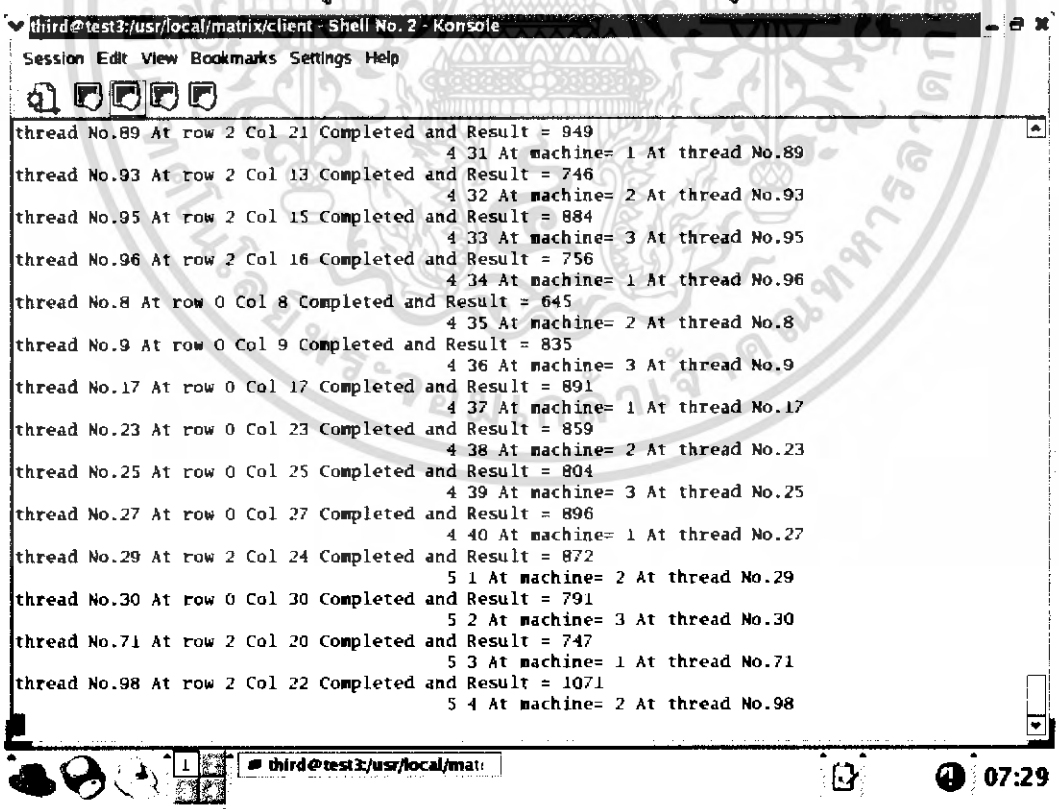
ตารางที่ 9-1 ตัวอย่างตารางบันทึกผลการทดลอง

กำหนดให้มีการทดสอบดังตารางที่ 9-1

9.4 ผลการทดลอง

9.4.1 ผลการทดลอง

เมื่อใช้คำสั่งรันโปรแกรมในรูปที่ 9-4 แล้วจะได้ผลการทดลองดังรูปที่ 9-5



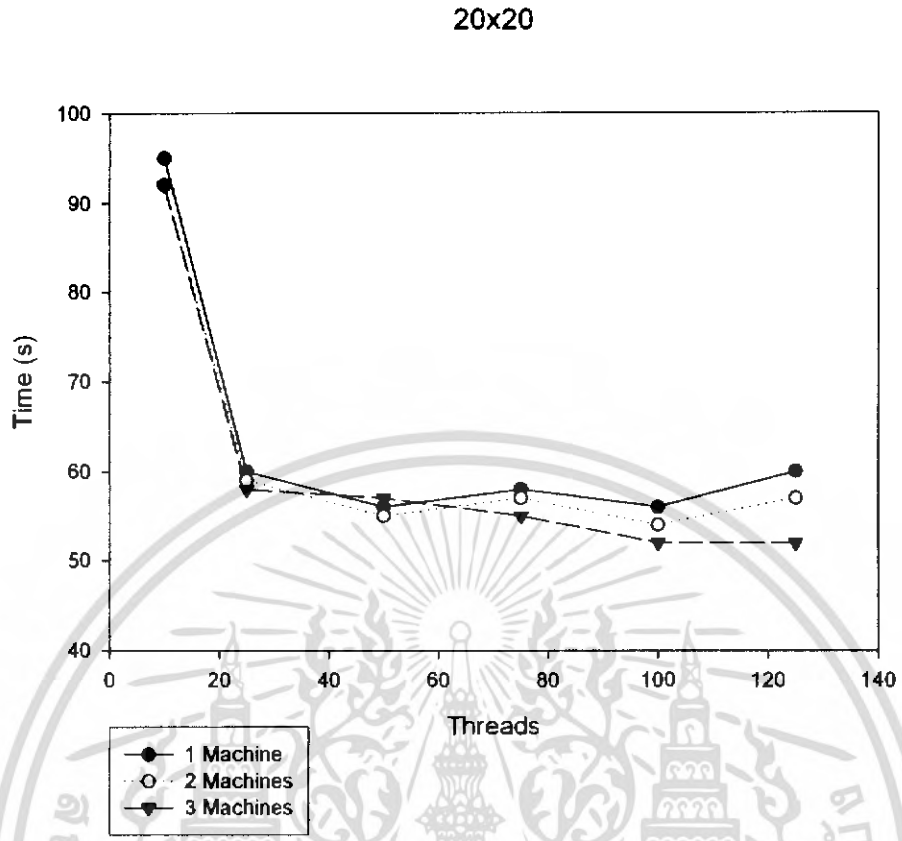
รูปที่ 9-5 ผลการแสดงขณะการทำงานของโปรแกรมในการแตกงาน

9.4.2 ตารางสรุปผลการทดลอง

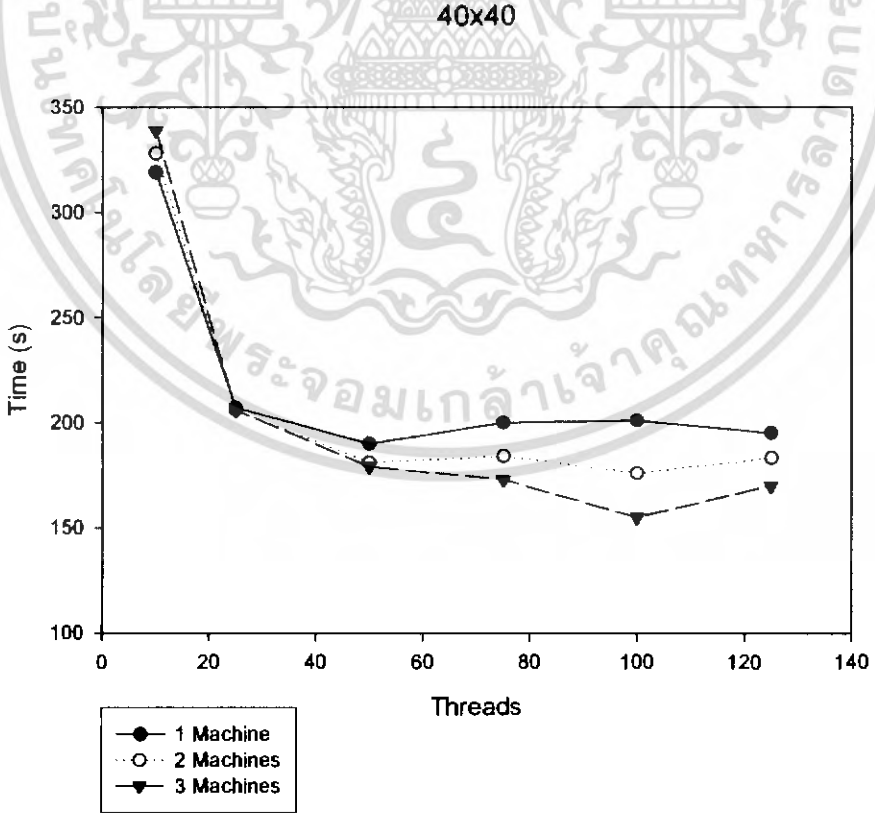
Thread	10			25			50			75			100			125		
	Number of machine			Number of machine			Number of machine			Number of machine			Number of machine			Number of machine		
Size	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
20*20	95	92	92	60	59	58	56	55	57	58	57	55	56	54	52	60	57	52
40*40	319	328	339	207	206	206	190	181	179	200	184	173	201	176	155	195	183	170
80*80	1269	1261	1205	766	754	739	635	628	625	747	726	618	760	748	624	731	719	696
160*160	5184	5040	4984	3003	2974	2943	2894	2833	2761	2955	2926	2901	3078	2872	2628	3150	3037	2941
320*320	18525	18315	18124	14567	13475	12393	12371	11984	11143	13521	13024	12451	14347	12974	12402	15423	14315	13116

ตารางที่ 9-2 แสดงผลการทดลองที่ได้ แสดงหน่วยเป็นวินาที

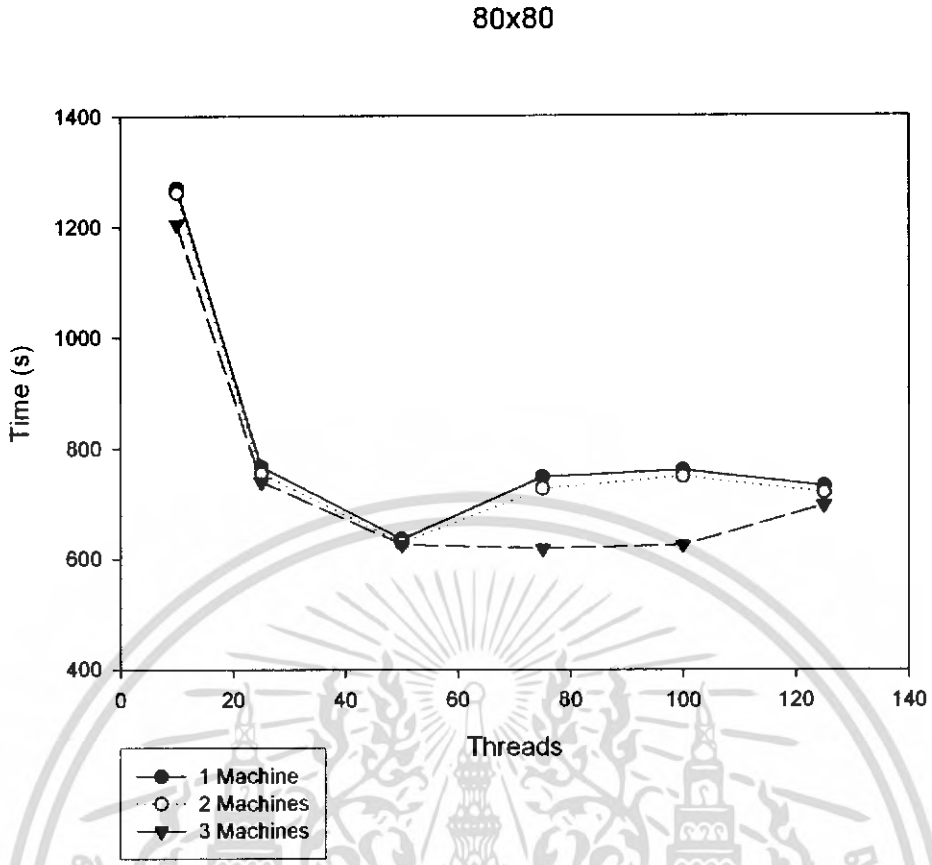
9.4.3 กราฟแสดงผลการทดลอง



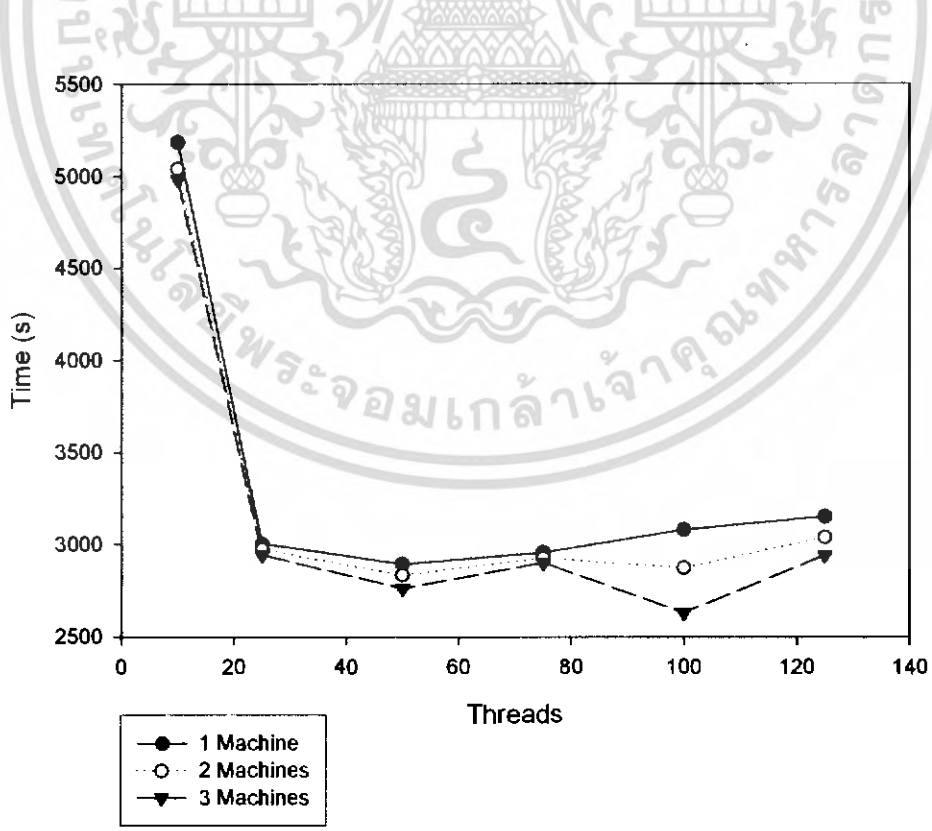
รูปที่ 9-6 ผลการทดลองของเมตริกขนาด 20*20



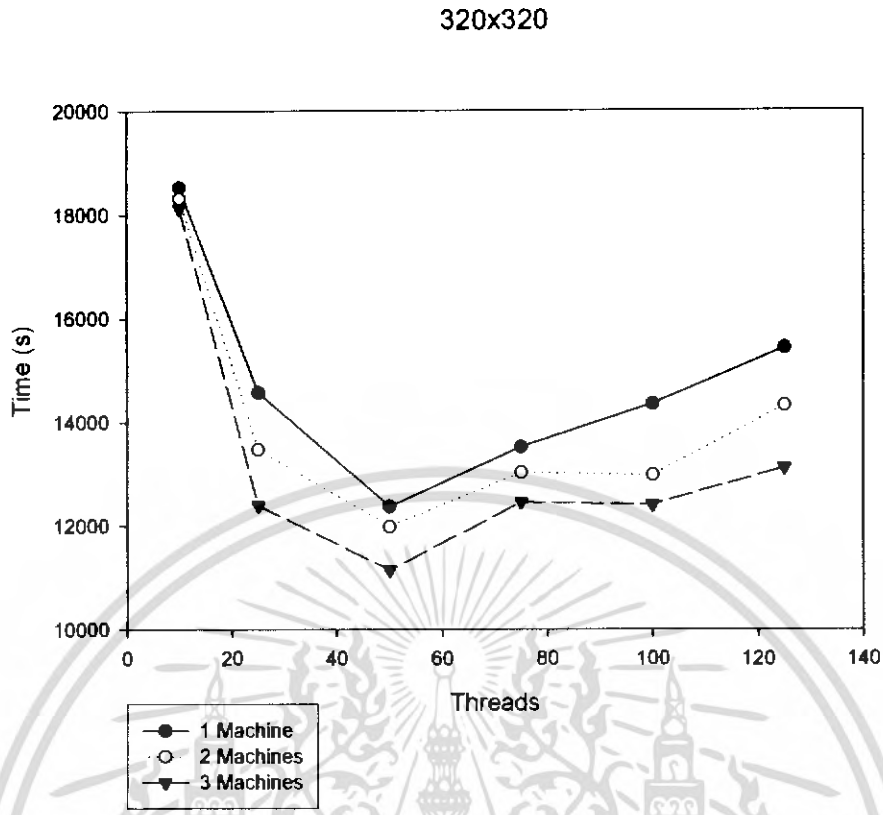
รูปที่ 9-7 ผลการทดลองของเมตริกขนาด 40*40



รูปที่ 9-8 ผลการทดลองของเมตริกขนาด 80*80



รูปที่ 9-9 ผลการทดลองของเมตริดขนาด 160*160



รูปที่ 9-10 ผลการทดลองของเมตริกขนาด 320*320