การสร้างระบบคลัสเตอร์แบบมีความเชื่อถือได้สูงและกระจายภาระด้วยลีนุกซ์

HIGH AVAILABILITY AND LOAD BALANCING USING LINUX CLUSTERING

นายจิรภัทร ฟาทอง

นายไชยศิริ เซ่งตระกูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การสร้างระบบคลัสเตอร์แบบมีความเชื่อถือได้สูงและกระจายภาระด้วยลีนุกซ์

HIGH AVAILABILITY AND LOAD BALANCING USING LINUX CLUSTERING

โดย

นายจิรภัทร พาทอง

นายไชยศิริ เซ่งตระกูล

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ดร.ชุติเมษฐ์ ศรีนิลทา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสร้างระบบคลัสเตอร์แบบมีความเชื่อถือได้สูงและกระจายภาระด้วยลีนุกซ์

HIGH AVAILABILITY AND LOAD BALANCING USING LINUX CLUSTERING

คณะผู้จัดทำ นายจิรภัทร พาทอง รหัส 43010064

นายไชยศิริ เช่งตระกูล รหัส 43010114

..... อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

..... อาจารย์ที่ปรึกษา

(ดร.ชุติเมษฐ์ ศรีนิลทา)

การสร้างระบบคลัสเตอร์แบบมีความเชื่อถือได้สูงและกระจายภาระด้วยลีนุกซ์

นายจิรภัทร พาทอง 43010064
นายไชยศิริ เช่งตระกูล 43010114
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ดร.ชุติเมษฐ์ ศรีนิลทา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ระบบคลัสเตอร์เป็นแนวทางหนึ่งในการแก้ปัญหาหลายๆ ปัญหา เช่น งานที่ต้องการพลังในการคำนวณที่ค่อนข้างสูง ก็นำระบบคลัสเตอร์แบบการประมวลผลแบบขนานมาใช้ ส่วนงานที่ต้องการประสิทธิภาพ และ ความเสถียรภาพของระบบที่สูง ก็นำระบบคลัสเตอร์แบบกระจายการทำงาน และ มีความเชื่อถือได้สูงมาใช้ ตัวอย่างของงานที่ต้องการประสิทธิภาพตลอดจนความเสถียรภาพของระบบที่สูง เช่น Web Server , FTP Server และ Mail Server ในปัจจุบันมีผู้ที่ใช้บริการผ่านเครือข่ายอินเตอร์เน็ตเป็นจำนวนมาก เพราะฉะนั้นการให้บริการเหล่านี้อาจต้องประสบปัญหาไม่สามารถรองรับสัญญาณร้องขอที่มีเข้ามามากเกินไปได้ ตลอดจนอาจเกิดเหตุสุดวิสัยที่ตัวเครื่องเซิร์ฟเวอร์เอง หนทางในการแก้ปัญหาเหล่านี้ คือ การใช้การกระจายงานไปยังเซิร์ฟเวอร์หลายๆ เครื่อง และทำให้แต่ละเครื่องมีเครื่องที่เป็นแบคอัพสามารถทำงานแทนเครื่องที่เกิดปัญหาได้ เพื่อที่จะสามารถรองรับสัญญาณการร้องขอบริการที่เป็นจำนวนมากๆ ได้และเพื่อป้องกันเหตุสุดวิสัยที่เกิดขึ้นกับตัวเซิร์ฟเวอร์ เช่น เครื่องขัดข้องขณะให้บริการ หรือการปิดบริการเพื่อทำการซ่อมบำรุง โดยขั้นตอนของการกระจายภาระนั้นจะใช้แนวคิดของ Linux Virtual Server และการทำให้ระบบมีความเชื่อถือได้สูงนั้นจะใช้แนวคิดของ High Availability เพื่อก่อเกิดประสิทธิภาพสูงสุดในการให้บริการ

HIGH AVAILABILITY AND LOAD BALANCING USING LINUX CLUSTERING

Mr. Chirapat Lathong
Mr. Chaisiri Sengtragul
Dr. Voravat Limpoka Advisor
Dr. Shutimet Srinilta Advisor
Academic Year 2003

ABSTRACT

Cluster System is one way to solve many problems such as job which need high power of calculation. It can be solved by using parallel processing clustering. On the other hand the job which need more efficiency and more reliability it can be solved by using high availability and load balancing clustering. System which need more efficiency and more reliability such as Web Server, Mail Server and FTP Server. Today too many users who using many services through the internet, so our service may be can not support user's need. It may be crash, hang or something that act like we can not give them our services. We may solve this problem by using high availability; give backup to host who running service. To distribute load to many real servers we will use The Linux Virtual Server method to lead to the highest efficiency of our services.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำ คำปรึกษาและคอยดูแลจากหลายๆ ฝ่าย ด้วยกัน โดยเฉพาะอาจารย์ที่ปรึกษาทั้งสองท่านที่ให้โอกาสให้ข้าพเจ้าได้ทำปริญญานิพนธ์ฉบับนี้ คอยให้ ความเอาใจใส่ แนะนำและความช่วยเหลือเสมอมา คือ อาจารย์วรวัฒน์ ลิ้มโภคา และ อาจารย์ชุติเมษฐ์ ศรีนิลทา ขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้การวิจัย และ พัฒนาเป็นไปด้วยความสะดวกและรวดเร็ว รวมทั้งยังมีอินเทอร์เนตความเร็วสูงให้บริการ สำหรับ การค้นคว้าหาความรู้ต่างๆ ซึ่งท้ายที่สุดแล้วก็ประกอบกันเป็นส่วนหนึ่งของโครงงานนี้

ขอขอบคุณพี่ๆ เพื่อนๆ น้องๆ ในห้องปฏิบัติการ OLALA ที่คอยสร้างความคึกครื้นยามอยู่ในห้อง เป็นกำลังใจเสมอมา

และสุดท้ายต้องขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดา และ บุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดูคอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาส ในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ ความรักเสมอมา ข้าพเจ้าขอกราบพระคุณมา ณ ที่นี้ด้วย

จิรภัทร พาทอง
ไชยศิริ เซ่งตระกูล

สารบัญ

หน้าที่

สารบัญ (ต่อ)

สารบัญตาราง

หน้าที่

สารบัญภาพ

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ระบบคลัสเตอร์ คือ การนำคอมพิวเตอร์หลายๆตัวมาเชื่อมต่อกันเพื่อให้ทำงานเหมือนเครื่องใหญ่เครื่องเดียว ความแตกต่างระหว่างระบบคลัสเตอร์กับระบบ LAN ก็คือ ระบบ LAN จะเป็นการเชื่อมต่อคอมพิวเตอร์จำนวนมากเพื่อใช้ทรัพยากรร่วมกันโดยแต่ละเครื่องยังเป็นเครื่องอิสระ แต่ระบบคลัสเตอร์เป็นการรวมเครื่องๆหลายหลายเครื่องที่ประสานงานกันอย่างแน่นแฟ้นจนเปรียบเสมือนเป็นเครื่องเดียวกัน ระบบคลัสเตอร์นี้สามารถนำมาประยุกต์ใช้งานได้แทนระบบเซิร์ฟเวอร์ขนาดใหญ่ได้เป็นอย่างดีในราคาที่ถูกกว่าอย่างไม่น่าเชื่อ

โครงสร้างของระบบพีซีคลัสเตอร์ทั่วไปจะประกอบด้วยเครื่องพีซีสมรรถนะสูงจำนวนหนึ่งที่นำมาเชื่อมต่อกันด้วยเครือข่ายความเร็วสูง เช่น ระบบ ATM Switch, Fast Ethernet Switch

โดยปกติโครงสร้างพื้นฐานของระบบคลัสเตอร์และระบบ LAN ไม่ต่างกันมาก แต่ระบบคลัสเตอร์จะพึ่งพาซอฟต์แวร์พิเศษที่กระจายงานไปในกลุ่มของคอมพิวเตอร์ที่มารวมกันทำงานเป็นระบบคลัสเตอร์ ดังนั้นคุณสมบัติสำคัญของเทคโนโลยีคลัสเตอร์มีอยู่สามประการ คือ เครือข่ายความเร็วสูง ระบบซอฟแวร์ที่สนับสนุนการทำงานแบบคลัสเตอร์และโปรแกรมประยุกต์ที่ใช้ขีดความสามารถดังกล่าว

ในปัจจุบันมีแนวโน้มทางเทคโนโลยีหลายประการที่สนับสนุนการก้าวไปสู่ระบบคลัสเตอร์ เช่น โพรเซสเซอร์ของเครื่องพีซีในปัจจุบันมีสมรรถนะแทบจะไม่แตกต่างจากโพรเซสเซอร์ของเครื่องซูเปอร์คอมพิวเตอร์ ดังนั้นการเชื่อมระบบพีซีเข้าด้วยกันจึงมีสมรรถนะไม่แตกต่างจากระบบซูเปอร์คอมพิวเตอร์ที่ขายกันอยู่เลย อย่างไรก็ตามเนื่องจากเทคโนโลยีของพีซีมีการขายจำนวนมหาศาล ผลที่ได้คือ ราคาที่ถูกมากเมื่อเทียบกับประสิทธิภาพ นอกจากนั้นการแข่งขันที่รุนแรงยังทำให้เงินทุนที่ทุ่มให้การวิจัยและพัฒนาเทคโนโลยีพีซีนั้นมหาศาล การพัฒนาจึงเป็นไปอย่างรวดเร็ว ทำให้ได้ของที่มีประสิทธิภาพสูงขึ้นมากในราคาเท่าเดิมตลอดเวลา

ปัญหาอย่างหนึ่งที่พบอยู่เสมอเมื่อใช้ระบบเซิร์ฟเวอร์ราคาแพง คือ ค่าบำรุงรักษาที่สูงมากเมื่อเทคโนโลยีนั้นเก่าหรือทำงานช้าไปแล้ว การเพิ่มระบบจะเป็นไปได้ยาก ในขณะที่ในระบบพีซีคลัสเตอร์สามารถเพิ่มความสามารถได้ทีละน้อยในราคาถูกกว่า ข้อดีประการสำคัญก็คือระบบพีซีเป็นเทคโนโลยีที่คนส่วนใหญ่คุ้นเคยทำให้สามารถบำรุงรักษาระบบได้โดยไม่ต้องจ้างบริษัทในราคาแพง

1.1.1 ลักษณะของคลัสเตอร์

เมื่อเราพูดถึงการคลัสเตอร์ คนส่วนมากจะนึกถึงแต่ว่าเป็นกลุ่มของคอมพิวเตอร์ที่มีประสิทธิภาพสูง อาจใช้ในงานวิจัยในงานวิทยาศาสตร์ อย่างไรก็ตามนั้นเป็นลักษณะของคลัสเตอร์ ซึ่งเราจะเรียกว่า Performance clustering ลักษณะของมันจะคล้ายกับว่าเครื่องคอมพิวเตอร์หลายๆเครื่องทำงานร่วมกันเหมือนเป็นเครื่องๆเดียวที่มีประสิทธิภาพสูง ซึ่งการทำคลัสเตอร์ ลักษณะนี้ มักจะถูกประยุกต์ใช้ในงานที่ต้องการจัดการปัญหาที่ซับซ้อน และต้องการพลังในการคำนวณที่ต้องสูงมากๆ อย่างเช่น การพยากรณ์อากาศ ดาราศาสตร์ เป็นต้น

ลักษณะที่สอง ของคลัสเตอร์ คือการทำให้เซิร์ฟเวอร์แต่ละตัวสามารถแบ่งโหลดและทราฟิก ที่เกิดขึ้นจากไคลเอนต์ได้ โดยการทำการกระจายภาระ (Load balancing) นี้เวลาในการเข้าถึง (access time) และ ความน่าเชื่อถือจะถูกปรับปรุงขึ้น จากการทำงานโดยใช้หลายๆ เซิร์ฟเวอร์ เมื่อมีตัวใดตัวหนึ่งเสียงานทั้งหมดย่อมจะไม่ล้มเหลวในคราวเดียว

ลักษณะสุดท้ายของคลัสเตอร์ คือการที่เรามีเซิร์ฟเวอร์ซึ่งทำหน้าที่เหมือนเป็นแบคอัพของตัวอื่น เราจะเรียกมันว่า High availability clustering (HA) หรือ Redundancy Clustering โดยการเฝ้าติดตามประสิทธิภาพ และ เสถียรภาพ ของเซิร์ฟเวอร์ตัวอื่นๆ และ HA สามารถเข้าทำงานในกรณีที่มีเซิร์ฟเวอร์อื่นๆ มีการล้มเหลว (failure) เกิดขึ้น

1.2 วัตถุประสงค์

1. สร้างระบบลีนุกซ์คลัสเตอร์ที่สามารถให้บริการ เช่น FTP Server หรือ Web Server ได้
2. เพื่อศึกษาการทำงานของการให้บริการ (Service) ต่างๆที่เปิดให้บริการได้
3. เพื่อศึกษาเทคโนโลยีทางด้านการคลัสเตอร์
4. สามารถทำการกระจายภาระ เพื่อเพิ่มประสิทธิภาพ ในการให้บริการ ต่างๆ
5. สามารถเปรียบเทียบข้อดีข้อเสียของแต่ละวิธีที่นำมาใช้ในการสร้างระบบ
6. สามารถทดสอบระบบที่ทำการติดตั้งได้

1.3 ขอบเขตของโครงงาน

ระบบที่สร้างขึ้นสามารถ ให้บริการ เช่น FTP หรือ Web Server และ Mail Serverได้ มีการแบ่งงาน ให้กับ Real Server แต่ละตัว โดยใช้ LVS รวมทั้งเครื่องที่เป็น HA สามารถ ทำงานเป็น Backup ของกันและกันได้ และทำการทดสอบระบบที่ได้ทำการติดตั้งได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถเข้าใจหลักการติดตั้งและใช้งาน FTP หรือ Web Server และ Mail Server ได้
2. สามารถเข้าใจถึงประโยชน์และหลักในการนำเทคโนโลยีทางด้านคลัสเตอร์มาใช้ร่วมกับการให้บริการ (Service) ที่เปิดได้ เพื่อให้ได้ระบบที่มีประสิทธิภาพ
3. สามารถเปรียบเทียบประสิทธิภาพในการทำงานแบบต่างๆที่ได้ทำการศึกษา

1.5 วิธีการดำเนินงาน

ขั้นตอนการดำเนินงานจะแบ่งออกเป็น 4 ขั้นตอนดังนี้คือ

1. ขั้นตอนการศึกษาหาข้อมูล

ทำการรวบรวมข้อมูลต่างๆที่จะนำมาใช้ในโครงงานนี้ ซึ่งจะเป็นเรื่องเกี่ยวกับเทคนิคทางด้านการคลัสเตอร์ รวมทั้งการติดตั้งและเปิดบริการ Service Web Server, FTP และ Mail Server ที่มีประสิทธิภาพซึ่งแหล่งที่ค้นคว้าก็จะมาจากหนังสือในห้องสมุด และจากอินเตอร์เน็ต

2. ขั้นตอนการติดตั้งระบบ

เป็นขั้นตอนของการติดตั้งการบริการต่างๆ รวมถึงโปรแกรมต่างๆที่ใช้ภายในโครงงานนี้

- การติดตั้งระบบปฏิบัติการลีนุกซ์ (ในโครงงานนี้เลือกใช้ ลีนุกซ์ Red Hat 8.0)
- การติดตั้ง Apache Web Server , FTP และ Mail Server บน Real Server
- ปรับแต่ง Kernel เพื่อรองรับแพคเกตที่จะนำมาติดตั้ง
- ติดตั้งแพคเกจต่างๆที่ใช้ในการนำมาใช้ในการทำงาน
- ทำการปรับเปลี่ยนระบบเพื่อให้ทำงานในแบบคลัสเตอร์ที่ต้องการ

3.ขั้นตอนทดลองและปรับแต่ง

เป็นขั้นตอนของการทำการทดลองและปรับแต่งประสิทธิภาพการทำงานของแต่ละการให้บริการให้มีประสิทธิภาพในการทำงานสูงสุด

4. ขั้นตอนการสรุปผล

เป็นขั้นตอนการสรุปผลทั้งหมดของโครงงาน โดยการเปรียบเทียบให้เห็นความแตกต่างของวิธีการต่างๆที่นำมาใช้ในการทำงาน อีกทั้งการทำงานที่เหมาะสมของวิธีการต่างๆที่ได้ทำการศึกษา รวมทั้งการปรับปรุงให้ระบบมีประสิทธิภาพมากขึ้น

บทที่ 2

Linux Virtual Server (LVS)

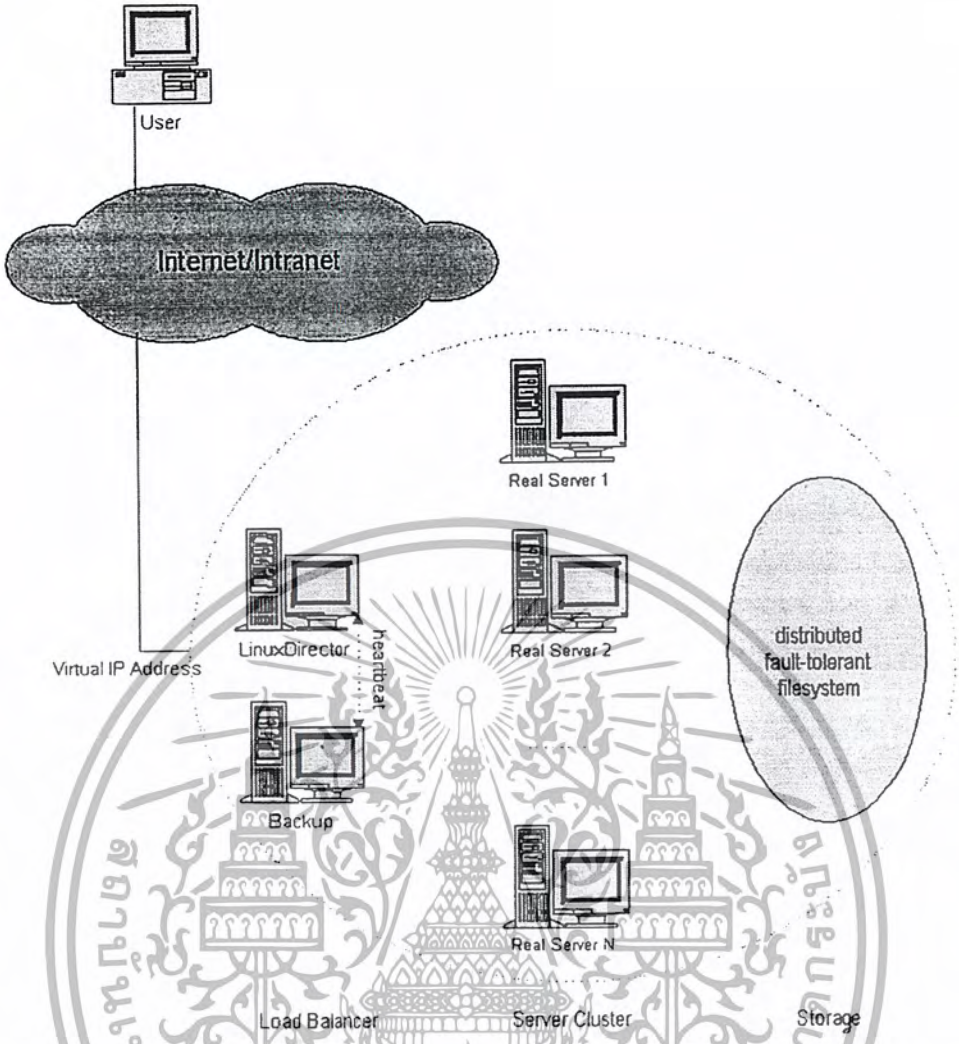
ระบบโดยรวม

The Linux Virtual Server Project (LVS) สามารถทำงานทางด้านการกระจายภาระ สำหรับการบริการทางระบบเครือข่ายได้ อย่างเช่น Web Server หรือ Mail Server โดยการใช้ Layer 4 Switching โดยจะทำให้การติดต่อเข้ามาขอใช้บริการ และการให้บริการเป็นไปได้อย่างรวดเร็ว และยังสามารถรองรับการเชื่อมต่อในเวลาเดียวกันได้ มากถึงหนึ่งแสนการเชื่อมต่อ

The Linux Virtual Server Project (LVS) นั้นสนับสนุน layer 4 switching ใน Linux Kernel เราจึงสามารถใช้ TCP และ UDP session ในการทำการกระจายภาระ ระหว่าง Real Server หลายๆ ตัวได้ และด้วยเหตุนี้เอง จึงเป็นการสนับสนุนการขยายขนาดของการบริการทางอินเทอเนตมากกว่าแบบที่มี Host เพียงตัวเดียว HTTP และ HTTPS ที่ใช้งานสำหรับ World Wide Web นั้น เป็นโพรโตคอลซึ่งส่วนใหญ่จะนำมาใช้งานกันบนระบบที่ทำงานแบบ LVS แม้นว่าตัว LVS นั้นจะทำงานอยู่บนระบบปฏิบัติการที่เป็น Linux แต่ก็ยังสามารถที่จะทำงานด้านการกระจายภาระ ให้กับ end users และ real server ที่ใช้ระบบปฏิบัติการที่เป็นอะไรก็ได้ ตราบใดที่การเชื่อมต่อดังกล่าวยังใช้งาน TCP หรือ UDP

2.1 LVS Basic

สถาปัตยกรรมโดยรวมของระบบ Linux Virtual Server ระบบ Linux Virtual Server จะมีสถาปัตยกรรมโดยรวมของระบบเป็นไปตามรูป 2-1



รูปที่ 2-1สถาปัตยกรรมโดยรวมของ Linux Virtual Server

จากรูปที่ 2-1 ระบบ Linux Virtual Server จะประกอบด้วย 3 ส่วนดังนี้คือ

- Load Balancer เป็นส่วนที่ติดต่อกับการร้องขอบริการทางอินเตอร์เน็ตจากไคลเอนต์ ที่มีเข้ามาในระบบ เครื่องโหลดบาลานซ์นี้จะมีหมายเลขประจำเครื่องที่สามารถติดต่อผ่านทางอินเตอร์เน็ตได้ และยังทำหน้าที่ในการส่งสัญญาณร้องขอบริการที่มีเข้ามาให้กับเครื่องเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์ให้ทำงานตามสัญญาณร้องขอบริการที่มีเข้ามา
- Server Cluster เป็นส่วนที่ประกอบด้วยเครื่องเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์ที่ทำหน้าที่ในการทำงานกับสัญญาณร้องขอบริการทางอินเตอร์เน็ตที่เครื่องโหลดบาลานซ์ส่งมาให้ทำงาน

- Storage เป็นส่วนที่ทำหน้าที่ในการเก็บข้อมูลโดยรวมของระบบ ซึ่งทำให้ระบบสามารถทำการจัดการกับข้อมูลที่ต้องให้บริการทางอินเตอร์เน็ตนั้นมีความเหมือนกันของเซิร์ฟเวอร์แต่ละเครื่องที่ต้องทำการบริการให้กับไคลเอนต์ที่ติดต่อเข้ามาผ่านทางเครื่องโหลดบาลานซ์

ขั้นตอนการทำงานของ Linux Virtual Server จะเป็นไปดังนี้คือ เครื่องโหลดบาลานซ์จะทำการจัดการกับสัญญาณการร้องขอบริการที่มีเข้ามาโดยการใช้วิธีการส่งต่อแพคเกตไปให้กับ Real Server โดยทำการเลือกเครื่องเซิร์ฟเวอร์เครื่องหนึ่งจากเครื่องเซิร์ฟเวอร์ทั้งหลายที่มีอยู่ในคลัสเตอร์ให้ทำงานกับสัญญาณการร้องขอบริการที่มีเข้ามา และเครื่องโหลดบาลานซ์ยังจะทำการรักษาสถานะในการติดต่อกับเครื่องไคลเอนต์ และพร้อมกับทำการส่งต่อ แพคเกต ไปให้กับเครื่องเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์ที่ได้ทำการเลือกไว้ ซึ่งงานทั้งหมดที่ทำที่เครื่องโหลดบาลานซ์นั้นจะทำภายใน kernel ของระบบปฏิบัติการ ดังนั้น overhead ของระบบนี้ในเครื่องโหลดบาลานซ์จะมีน้อย ส่งผลให้เครื่องโหลดบาลานซ์สามารถจัดการกับสัญญาณร้องขอที่มีเข้ามาได้มากกว่าเซิร์ฟเวอร์ทั่วไป ซึ่งจะทำให้ประสิทธิภาพในการทำงานโดยรวมของระบบดีขึ้นมาก โดยระบบสามารถรองรับกับสัญญาณการร้องขอบริการทางอินเตอร์เน็ตที่มากขึ้นได้อย่างมีประสิทธิภาพ

เครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ที่ทำงานตามที่เครื่องโหลดบาลานซ์ส่งมาให้ทำนั้นจะทำให้เกิดความสามารถที่เรียกว่า Scalability โดยที่การ Scalability นั้นจะทำได้โดยการเพิ่มหรือลดเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ได้โดยตรงตามความต้องการใช้งาน เช่น เมื่อภาระงานที่ระบบต้องทำงานมีมากขึ้นเกินความสามารถของระบบที่จะรองรับได้ เราก็จะทำการเพิ่มเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์เพื่อให้ระบบสามารถรองรับกับภาระงานที่มีมากขึ้นได้ ซึ่งประสิทธิภาพของระบบนั้นจะเพิ่มมากขึ้นตามจำนวนเครื่องเซิร์ฟเวอร์ที่มีอยู่ในคลัสเตอร์

ข้อได้เปรียบอีกประการหนึ่งของระบบคลัสเตอร์ก็คือมันสามารถทำการ Redundancy ได้กับทั้งฮาร์ดแวร์และซอร์ฟแวร์ ซึ่งทำให้มันมีความสามารถที่เรียกว่า Availability โดยความสามารถที่ว่านี้ก็คือระบบจะทำการตรวจจับความผิดพลาดที่แต่ละเครื่องเซิร์ฟเวอร์และเมื่อมันพบว่ามีความผิดพลาดเกิดขึ้นมันก็จะทำการปรับเปลี่ยนระบบได้อย่างเหมาะสมโดยอัตโนมัติ ซึ่งระบบจะทำการส่งภาระงานไปให้กับเครื่องเซิร์ฟเวอร์ที่เหลือภายในคลัสเตอร์ให้ทำงานแทนได้ โดยวิธีการนี้เราจะมีโปรแกรม monitor daemon ทำงานอยู่ที่เครื่องโหลดบาลานซ์ซึ่งจะทำการตรวจสอบดูสภาพการทำงานของระบบคลัสเตอร์โดยจะตรวจสอบว่าเซิร์ฟเวอร์แต่ละเครื่องภายในคลัสเตอร์นั้นยังทำงานอยู่หรือไม่ ซึ่งถ้าเครื่องเซิร์ฟเวอร์เครื่องหนึ่งเครื่องใดไม่ทำการตอบรับกับสัญญาณ ICMP ping หรือไม่มีการตอบรับกับสัญญาณร้องขอบริการที่เครื่องโหลดบาลานซ์ส่งไปให้ทำงานภายในช่วงระยะเวลาหนึ่ง โปรแกรม monitor ที่เครื่องโหลดบาลานซ์ก็จะทำให้เซิร์ฟเวอร์เครื่องนั้นไม่ได้ถูกใช้งานภายในระบบคลัสเตอร์ โดยการลบเซิร์ฟเวอร์เครื่องนั้นออกจาก scheduling table ที่อยู่ในเครื่องโหลดบาลานซ์ ซึ่งก็จะส่งผลให้เครื่องโหลดบาลานซ์ไม่ทำการส่งสัญญาณการร้องขอบริการที่มีเข้ามาให้กับเครื่องเซิร์ฟเวอร์ที่ทำงานผิดพลาดเครื่องนั้นภายในคลัสเตอร์ทำงานได้

เครื่องโหลดบาลานซ์อาจจะทำให้ทั้งระบบไม่สามารถทำงานได้เมื่อเครื่องโหลดบาลานซ์มีข้อผิดพลาดเกิดขึ้นและได้หยุดการทำงานหรือที่เรียกว่า Single Point of Failure เพื่อที่จะทำให้ระบบยังทำงานต่อไปได้แม้ว่าเครื่องโหลดบาลานซ์จะหยุดการทำงานอันเนื่องมาจากสาเหตุใดก็ตาม เราสามารถที่จะทำการแก้ไขได้โดยการติดตั้งเครื่องที่ทำหน้าที่แทนเครื่องโหลดบาลานซึ่งที่สามารถทำงานได้โดยเราจะเรียกเครื่องนี้ว่าเครื่องสำรอง (backup) ซึ่งวิธีการก็คือเราจะใช้โปรแกรม daemon ที่ทำหน้าที่ในการตรวจสอบสถานการณ์การทำงานของอีกเครื่องหนึ่งว่ายังคงทำงานอยู่หรือไม่ โดยจะให้โปรแกรม daemon นี้ทำงานอยู่ที่เครื่องที่ทำหน้าที่เป็นโหลดบาลานซ์ของระบบกับเครื่องที่ทำหน้าที่สำรอง ซึ่งโปรแกรม daemon นี้จะทำการตรวจสอบสถานการณ์ทำงานผ่านทางสายแบบอนุกรม (serial line) หรือผ่านทาง UDP ในทุกๆช่วงเวลา เมื่อ daemon ที่เครื่องสำรองไม่ได้รับสัญญาณตอบรับจากเครื่องโหลดบาลานซ์ในช่วงเวลาที่กำหนดไว้ เครื่องสำรองก็จะทำการใช้วิธี ARP spoofing เพื่อทำให้เครื่องสำรองเปลี่ยนหมายเลขไอพีมาเป็นหมายเลขที่ไอพีที่ให้บริการทางอินเตอร์เน็ตของเครื่องโหลดบาลานซ์ที่ทำหน้าที่รับสัญญาณการร้องขอบริการจากอินเตอร์เน็ต ต่อจากนั้นเครื่องสำรองก็จะทำหน้าที่แทนเครื่องโหลดบาลานซ์ และเมื่อเครื่องที่เป็นโหลดบาลานซ์ที่หยุดการทำงานไปกลับมาทำงานได้เหมือนเดิม การทำงานจะเป็นไปในสองวิธีนี้คือ

1. เครื่องที่เป็นโหลดบาลานซ์ที่กลับมาทำงานได้เหมือนเดิมจะกลายมาเป็นเครื่องสำรองให้กับเครื่องสำรองที่ทำหน้าที่แทนเครื่องโหลดบาลานซ์
2. เมื่อโปรแกรม daemon ที่เครื่องสำรองที่ทำหน้าที่แทนเครื่องโหลดบาลานซ์ได้รับสัญญาณตอบรับจากเครื่องทีเป็นโหลดบาลานซ์ที่กลับมาทำงานได้เหมือนเดิม เครื่องสำรองก็จะทำการคืนหมายเลขไอพีที่เป็นหมายเลขที่ใช้ในการบริการทางอินเตอร์เน็ตให้กับเครื่องโหลดบาลานซ์ที่กลับมาทำงานได้ใหม่

อย่างไรก็ตามทั้งสองวิธีนี้ในขั้นตอนที่เครื่องโหลดบาลานซ์เกิดทำงานผิดพลาดและเครื่องสำรองทำงานแทน จะทำให้การติดต่อของสัญญาณการร้องขอบริการก่อนหน้านั้นหายไป ซึ่งส่งผลให้ไคลเอนต์ต้องทำการส่งสัญญาณร้องขอบริการเข้ามาใหม่

จากรูปที่ 2-1 นั้นส่วนที่เป็น Backend Storage โดยปกติแล้วจะเป็นระบบไฟล์ที่คงทนตอความผิดพลาด(distributed fault-tolerant file system) ตัวอย่างเช่น GFS , Coda หรือ Intermezzo ซึ่งระบบไฟล์เหล่านี้จะคำนึงถึงเรื่อง Availability กับ Scalability ของการเข้าถึงไฟล์ของระบบ ซึ่งเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์จะทำการเข้าถึงระบบไฟล์เหมือนกับการเข้าถึงไฟล์ที่เครื่องของตนเอง อย่างไรก็ตามโปรแกรมประยุกต์ที่เป็นโปรแกรมเดียวกันจำนวนหลายโปรแกรมที่ทำงานภายในเซิร์ฟเวอร์แต่ละเครื่องภายในคลัสเตอร์อาจจะทำการเข้าถึงทรัพยากรของระบบเดียวกันแบบพร้อมกัน ซึ่งจะส่งผลให้เกิดความขัดแย้ง (conflict) โดยปกติแล้วโปรแกรมประยุกต์แต่ละโปรแกรมจะต้องทำให้ไม่เกิดการขัดแย้งเกิดขึ้นเมื่อเข้าใช้งานทรัพยากรของระบบพร้อมกันเพื่อให้ทรัพยากรนั้นอยู่ในสถานะที่เสถียรซึ่งจำเป็นต้องมีโปรแกรมที่ทำหน้าที่นี้ (distributed lock manager) ซึ่งอาจจะอยู่ในระบบไฟล์ที่ใช้หรืออาจจะอยู่ภายนอกระบบไฟล์ที่ใช้ก็ได้โปรแกรมจัดการนี้จะทำให้ผู้พัฒนาโปรแกรมประยุกต์สามารถที่จะทำการโปรแกรม

ได้อย่างง่ายกับการทำงานพร้อมกันกับการเข้าถึงทรัพยากรของระบบในโปรแกรมประยุกต์ที่ทำงานอยู่ในแต่ละเครื่องเซิร์ฟเวอร์ภายในระบบคลัสเตอร์

2.2 คำศัพท์ที่ควรรู้

- Linux Director (Load Balancer) : Host ซึ่งเราได้ทำการติดตั้ง Linux และ LVS ลงไป จะทำหน้าที่รับแพคเกตจาก end users และ ส่งต่อแพคเกตเหล่านั้นให้กับ real servers
- Real Server : Host ซึ่งเป็นจุดสิ้นสุดของการเชื่อมต่อ จะทำหน้าที่รัน service ต่างๆ
- Virtual IP Address (VIP) : เป็น IP address ที่ end users มองเห็นและใช้เป็นช่องทางในการติดต่อขอใช้บริการจากระบบ ซึ่งตัว Linux Director จะเป็นผู้จัดการกับ VIP นี้
- Real IP Address (RIP) : เป็น IP address ของ Real Server

2.3 Layer 4 Switching

Layer 4 Switching ทำงานโดยการ multiplexing TCP/IP และ UDP/IP ที่เข้ามาส่งไปให้กับ Real Server แพคเกตจะถูกรับโดย Linux Director และ จะทำการตัดสินใจว่า Real Server ตัวใดที่จะทำการส่งแพคเกตเหล่านั้นไปให้ และ จากการเลือก Real Server ขึ้นมาทำงานนั้น การส่งแพคเกตต่อๆ มาที่เป็นของการเชื่อมต่อนั้นๆ ก็จะต้องส่งไปที่ Real Server ตัวเดียวกันด้วย

2.4 Forwarding Packets

ในการติดตั้งระบบนี้เครื่องที่ทำหน้าที่เป็น Linux Director จะต้องทำการเพิ่มวิธีการเหล่านี้ (patch) เข้าไปใน Kernel ของระบบปฏิบัติการ ซึ่งจะทำให้เครื่อง Linux Director ทำหน้าที่ในการติดต่อกับสัญญาณการร้องขอบริการของเครื่องไคลเอนต์ โดยที่เครื่องไคลเอนต์จะรู้เพียงหมายไอพีของเครื่อง Linux Director เท่านั้น จากนั้นเครื่อง Linux Director จะทำการส่งสัญญาณการร้องขอการใช้บริการให้กับ Real Server เครื่องใดเครื่องหนึ่งในคลัสเตอร์ ซึ่ง Linux Virtual Server นั้นสนับสนุนการ forwarding packets ด้วยกัน 3 วิธีได้แก่

1. Network Address Translation (NAT)
2. Direct Routing
3. Tunneling (IP-IP Encapsulation)

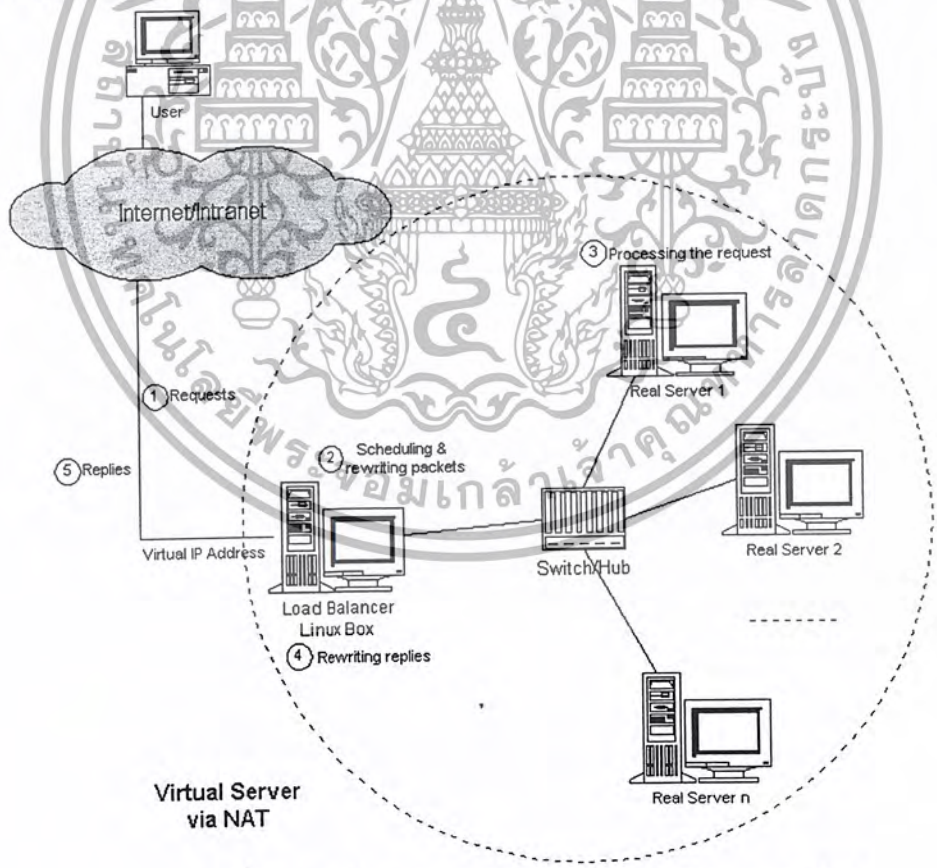
2.4.1 Network Address Translation (NAT)

เนื่องมาจากความขาดแคลนหมายเลขไอพีและปัญหาด้านความปลอดภัยใน IPv4 เครือข่ายต่างๆ จำนวนมากได้หันมาใช้หมายเลขไอพีภายในที่ไม่สามารถใช้งานได้ทางอินเตอร์เน็ต เช่น 0.0.0.0/255.0.0.0 เป็นต้น หลักการ Network Address Translation จึงเกิดขึ้นเพื่อให้เครื่องในเครือข่ายภายในสามารถติดต่อออกทางอินเตอร์เน็ตได้ และสามารถเข้าถึงได้จากเครื่องที่อยู่ในอินเตอร์เน็ต หลักการของ NAT จะอาศัยความจริงที่ว่า header ของ แพคเกต สามารถที่จะทำการเปลี่ยนแปลงอย่างเหมาะสมตามความต้องการใช้

งาน ซึ่งจะส่งผลให้ไคลเอนต์เชื่อว่าตนเองนั้นกำลังทำการติดต่ออยู่กับหมายเลขประจำเครื่องของเครื่องที่ให้บริการทางอินเตอร์เน็ตเพียงหมายเลขเดียว และเครื่องเซิร์ฟเวอร์ต่างๆ ที่อยู่ภายในคลัสเตอร์นั้น แต่ละเครื่องก็จะมีหมายเลขประจำเครื่องของมันเองซึ่งแต่ละเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์จะเชื่อว่าตนเองกำลังทำการติดต่อกับไคลเอนต์โดยตรง ซึ่งลักษณะรูปแบบนี้สามารถที่จะนำมาสร้าง Virtual Server ตัวอย่างเช่น การบริการแบบขนานที่มีเครื่องเซิร์ฟเวอร์หลายเครื่องที่มีหมายเลขประจำเครื่องเป็นของตนเองสามารถที่จะนำมาทำเป็นการบริการแบบ Virtual Service ที่ทำงานโดยผ่านหมายเลขไอพีเพียงหมายเลขเดียวได้

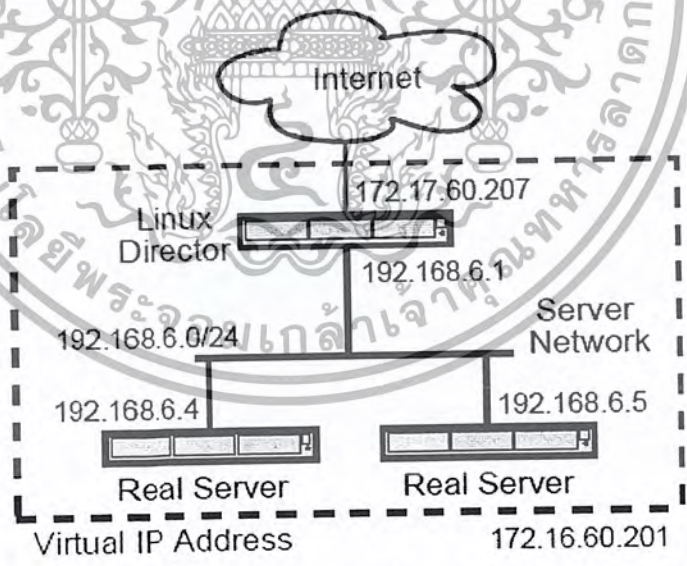
วิธีการทำงานของ Network Address Translation จะเป็นการจับคู่หมายเลขไอพีจากกลุ่มหนึ่งไปยังหมายเลขไอพีอีกกลุ่มหนึ่ง เมื่อการจับคู่เป็นแบบ N-to-N วิธีการนี้จะเรียกว่า Static Network Address Translation และถ้าการจับคู่เป็นแบบ M-to-N (M>N) จะเรียกวิธีการนี้ว่า Dynamic Address Translation และหลักการของ Network Address Port Translation จะมีวิธีการมาจากพื้นฐานของ NAT ซึ่งหมายเลขไอพีจำนวนหนึ่งและหลายหมายเลข port ใน TCP/UDP จะถูกแปลงเป็นหมายเลขไอพีเดียวและหลายหมายเลข port วิธีการแบบนี้จะเป็นการจับคู่แบบ N-to-1 ซึ่งจะเป็นหลักการที่ Linux IP Masquerading นำมาใช้

Virtual Server via NAT จะใช้หลักการของ Network Address Port Translation ในการทำงาน ซึ่งจะอาศัยจากการทำงานของ Linux IP Masquerading ที่มีอยู่ในลินุกซ์



รูปที่ 2-2 สถาปัตยกรรมของระบบ Linux Virtual Server via NAT

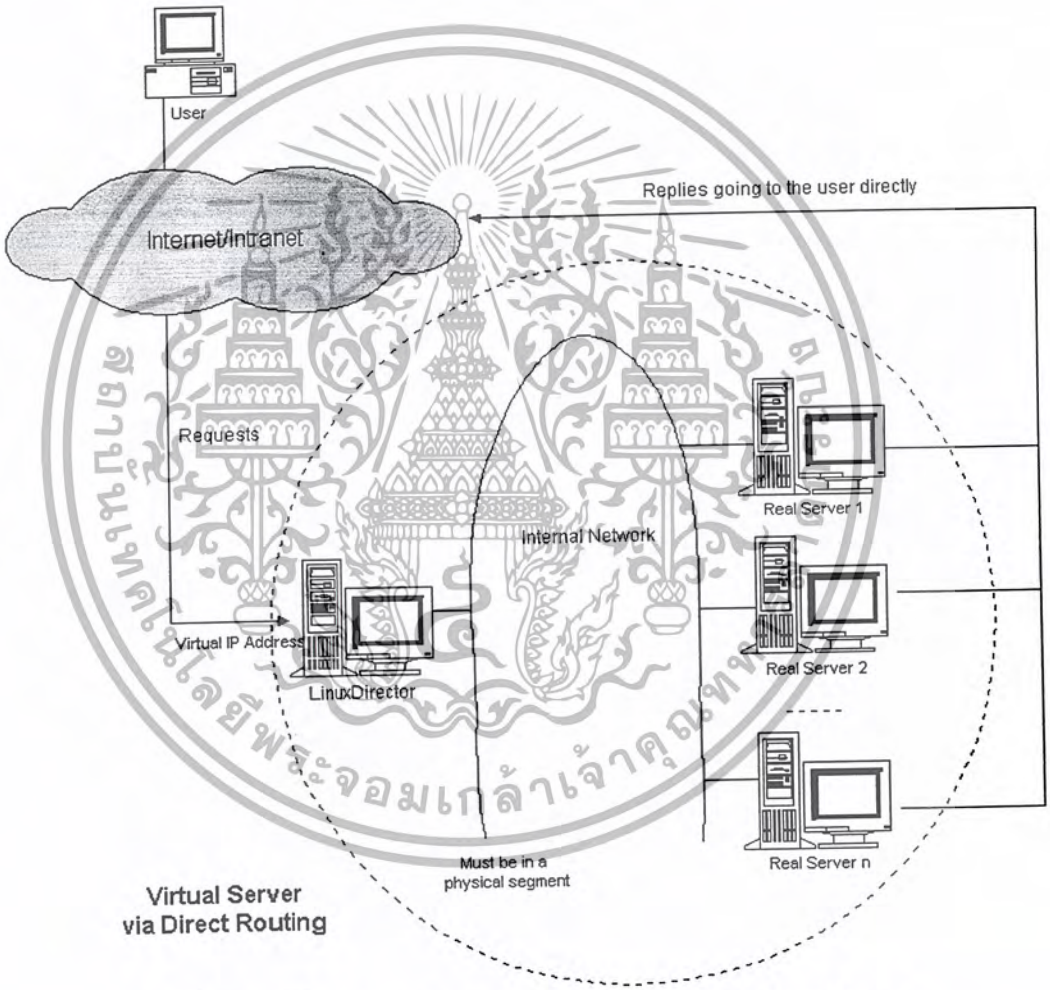
สถาปัตยกรรมของระบบ Linux Virtual Server via NAT จะแสดงให้เห็นตามรูปที่ 2-2 โดยที่เครื่องโหลดบาลานซ์และเครื่องเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์จะทำการต่อกันโดยใช้ switch หรือ hub ซึ่งขั้นตอนการทำงานของ Virtual Server via NAT จะทำงานเป็นไปดังนี้ เมื่อผู้ใช้ได้ทำการส่งสัญญาณร้องขอบริการมาที่ระบบคลัสเตอร์ที่ทำการบริการแบบ virtual service ซึ่งสัญญาณร้องขอที่มีเข้ามานั้นจะมีที่อยู่ปลายทางเป็นหมายเลขไอพีที่ทำการบริการทางอินเตอร์เน็ต (virtual IP address) ซึ่งสัญญาณร้องขอนี้จะเข้ามาที่ระบบคลัสเตอร์โดยผ่านเครื่องโหลดบาลานซ์ ต่อจากนั้นเครื่องโหลดบาลานซ์จะทำการตรวจสอบดูว่าหมายเลขไอพีและหมายเลข port ว่าเป็นของ virtual IP address หรือไม่ ถ้าใช่เครื่องโหลดบาลานซ์จะทำการเลือกเซิร์ฟเวอร์ที่จะทำการประมวลผลภายในคลัสเตอร์โดยวิธีการเลือกนั้นจะเป็นไปตาม Scheduling Algorithm ที่ได้เลือกไว้และการติดต่อจะถูกเก็บไว้ใน hash table ซึ่งจะใช้สำหรับบันทึกการติดต่อที่มีเข้ามา จากนั้นหมายเลขไอพีปลายทางกับหมายเลข port ของ แพคเกต ที่มีเข้ามาจะถูกเขียนขึ้นใหม่ให้เป็นหมายเลยไอพีและหมายเลข port ของเครื่องเซิร์ฟเวอร์ที่ได้ถูกเลือกให้ทำงานกับสัญญาณร้องขอที่มีเข้ามานั้น และจากนั้นก็จะทำการส่ง แพคเกต ไปให้กับเซิร์ฟเวอร์เครื่องนั้น และเมื่อการติดต่อที่มีเข้ามาเป็นการติดต่อที่ได้สร้างขึ้นแล้ว ซึ่งได้ทำการบันทึกอยู่ใน hash table การทำงานก็จะทำโดยการเขียน header ของ แพคเกต ใหม่และทำการส่งไปให้กับเซิร์ฟเวอร์ที่ได้ทำการบันทึกใน hash table และเมื่อ แพคเกต ตอบรับได้ทำการส่งกลับมาให้กับเครื่องโหลดบาลานซ์ ซึ่งเครื่องโหลดบาลานซ์ก็จะทำการเขียนหมายเลขไอพีและหมายเลข port ต้นทางใหม่ให้เป็นหมายเลขของ Virtual IP Address เมื่อการติดต่อนั้นจบการทำงานหรือเกิดการ timeout สิ่งที่ได้บันทึกใน hash table ก็จะถูกลบทิ้งไป
ตัวอย่างที่แสดงการทำงานโดยละเอียดของวิธีการแบบ NAT จะเป็นไปตามการติดตั้งในรูป



รูปที่ 2-3 ตัวอย่างการติดตั้งของระบบ Linux Virtual Server via NAT

2.4.2 Direct Routing

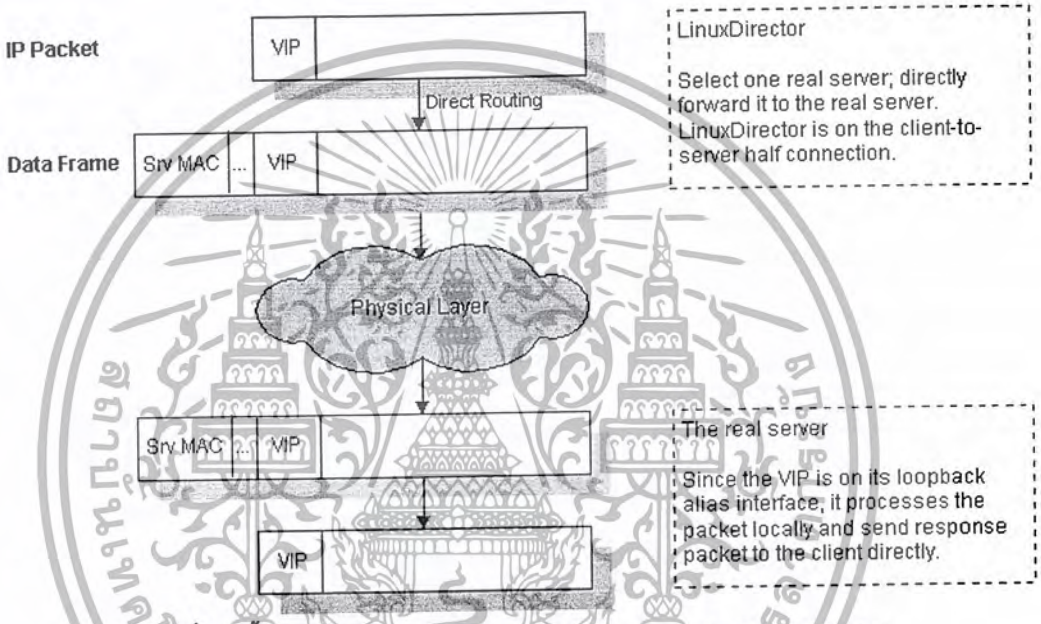
วิธีนี้จะคล้ายกันกับวิธีการที่ใช้สร้าง Net Dispatcher ของบริษัท IBM สถาปัตยกรรมของระบบ Linux Virtual Server via Direct Routing จะแสดงอยู่ในรูปที่ 2-4 โดยเครื่องที่ทำหน้าที่เป็นโหลดบาลานซ์และเครื่องเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์จะทำการต่อกันใน LAN วงเดียวกัน เช่น ต่อกันโดย Hub หรือ โดย Switch โดยที่เครื่องโหลดบาลานซ์และเครื่องเซิร์ฟเวอร์ต่างๆภายในคลัสเตอร์จะใช้หมายเลขไอพีที่เป็น Virtual IP ร่วมกัน โดยที่เซิร์ฟเวอร์ทุกเครื่องที่อยู่ภายในคลัสเตอร์จะมีการใช้อินเตอร์เฟสเหมือนแบบ loopback ที่ทำการติดตั้งเป็น Virtual IP และเครื่องโหลดบาลานซ์จะมีการใช้อินเตอร์เฟสที่ทำการติดตั้งเป็น Virtual IP เพื่อทำการรับกับสัญญาณการร้องขอบริการที่มีเข้ามาในระบบคลัสเตอร์



รูปที่ 2-4 สถาปัตยกรรมของระบบ Linux Virtual Server via Direct Routing

ขั้นตอนการทำงานของระบบคลัสเตอร์แบบ Linux Virtual Server via Direct Routing นั้นจะทำงานเหมือนกับ Linux Virtual Server via NAT กับ Linux Virtual Server via IP Tunneling โดยความ

แตกต่างจะอยู่ที่เครื่องโหลดบาลานซ์เมื่อได้รับสัญญาณการร้องขอบริการทางอินเตอร์เน็ตแล้วจะทำการเปลี่ยนที่อยู่ของ MAC (MAC address) ใน data frame ให้เป็นของเครื่องเซิร์ฟเวอร์ที่ได้ทำการเลือกภายในคลัสเตอร์ และจากนั้นจะทำการส่งแพคเกต นั้นไปตามเครือข่าย LAN เมื่อเครื่องเซิร์ฟเวอร์ที่ได้เลือกได้รับ แพคเกต เครื่องเซิร์ฟเวอร์เครื่องนั้นก็จะพบว่า แพคเกต นั้นมีที่อยู่ปลายทางเป็นอินเตอร์เฟสเสมือนของ loopback ของเครื่องมันเองและจากนั้นมันก็จะทำงานกับสัญญาณการร้องขอบริการ และจากนั้นก็จะทำการคืนผลลัพธ์ไปให้กับไคลเอนต์โดยตรงโดยที่เครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์แต่ละเครื่องนั้นจะมีอินเตอร์เฟสที่ทำการติดตั้งเป็น Virtual IP ที่จะไม่ทำการ ARP response มิฉะนั้นแล้วจะเกิดการชนกันของหมายเลขไอพีเพราะว่ามีหมายเลขไอพีซ้ำกันอยู่ภายในเครือข่ายเดียวกัน วิธีการของ Direct Routing จะมีขั้นตอนการทำงานตามรูปที่ 2-5



รูปที่ 2-5 ขั้นตอนการทำงานของ Linux Virtual Server via Direct Routing

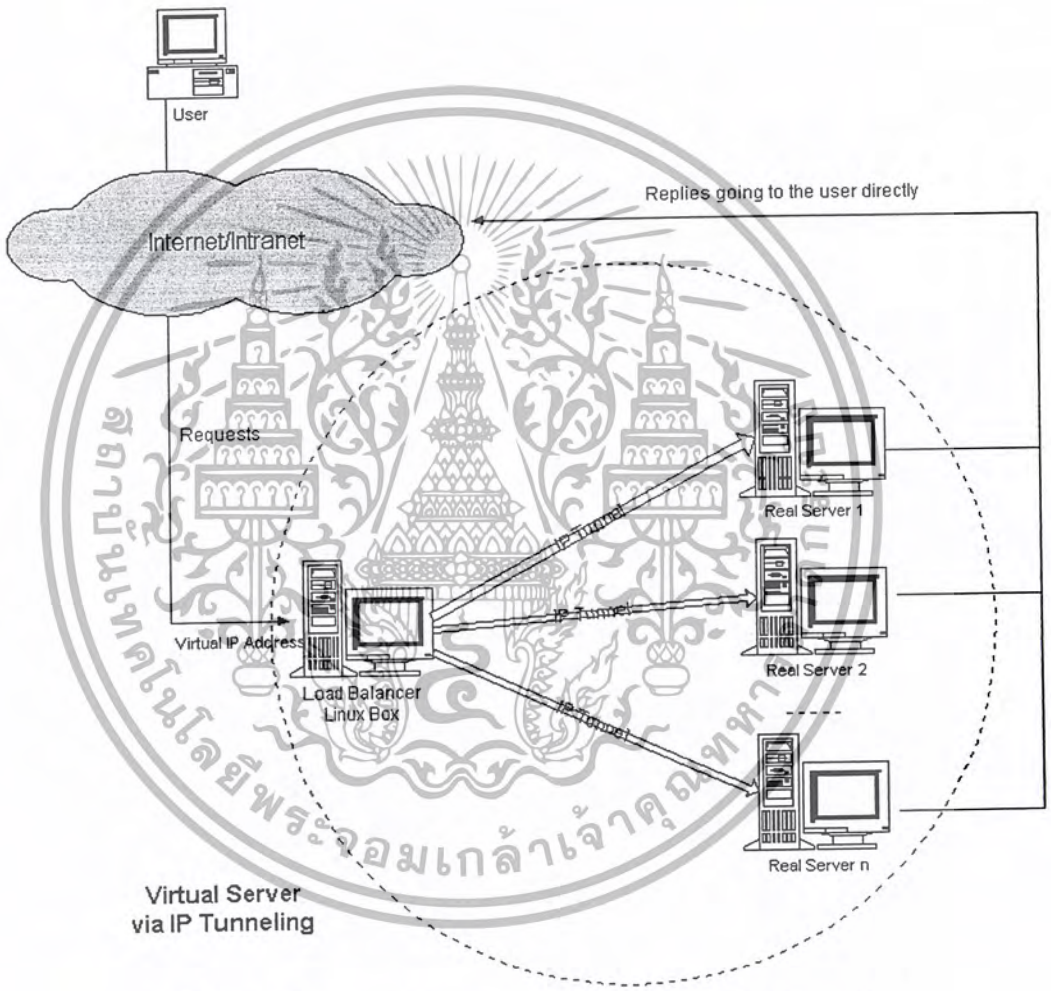
จากรูปที่ 2-5 เครื่องโหลดบาลานซ์จะทำการเปลี่ยน MAC address ของ data frame ในสัญญาณการร้องขอบริการเป็นของเครื่องเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์และทำการส่ง แพคเกต นั้นไปให้กับไคลเอนต์โดยตรงผ่านทาง LAN ซึ่งเป็นเหตุผลที่ทำให้เครื่องโหลดบาลานซ์และเครื่องเซิร์ฟเวอร์ต่างๆ ภายในคลัสเตอร์จำเป็นที่ต้องอยู่ใน LAN วงเดียวกัน

2.4.3 Tunneling (IP-IP Encapsulation)

วิธีการของ IP Tunneling เป็นเทคนิคที่ทำการ encapsulate กับ IP datagram ภายใน IP datagram อีกทีหนึ่ง ซึ่งจะทำให้ datagram ที่มีที่อยู่ปลายทางเป็นหมายเลขไอพีหนึ่งจะถูกทำการ encapsulate เป็นหมายเลขไอพีอีกหมายเลขหนึ่งและทำการ redirect ไปให้กับหมายเลขไอพีที่ทำการ encapsulate นั้น ซึ่งเทคนิคนี้สามารถนำมาใช้สร้าง Virtual Server ซึ่งเครื่องที่เป็นโหลดบาลานซ์จะทำการ tunnel กับสัญญาณร้องขอที่มีเข้ามาไปให้กับเซิร์ฟเวอร์เครื่องต่างๆภายในคลัสเตอร์ และเครื่องเซิร์ฟเวอร์

ภายในคลัสเตอร์นั้นจะทำงานกับสัญญาณการร้องขอบริการที่เครื่องโหลดบาลานซ์ส่งมาให้ จากนั้นก็จะทำการส่งผลลัพธ์ไปให้กับไคลเอนต์โดยตรง ซึ่งจะทำให้บริการที่ระบบให้กับไคลเอนต์เป็นแบบ Virtual Server ที่ใช้หมายเลขไอพีที่ให้บริการทางอินเตอร์เน็ตเพียงหมายเลขเดียว

สถาปัตยกรรมของ Virtual Server via IP Tunneling จะถูกแสดงอยู่ในรูป 2-6 โดยเครื่องที่เป็นเซิร์ฟเวอร์ภายในคลัสเตอร์นั้นสามารถมีไอพีจริงที่สามารถใช้ได้ในอินเตอร์เน็ต ซึ่งสามารถอยู่ที่เครือข่ายไหนก็ได้สักแห่ง แต่ว่าเครื่องเซิร์ฟเวอร์เหล่านั้นจะต้องทำการสนับสนุนกับโพรโตคอล IP Tunneling และที่เครื่องเซิร์ฟเวอร์เหล่านั้นจำเป็นที่จะต้องมีอินเตอร์เฟสที่ทำการติดตั้งเป็น Virtual IP หรือ หมายเลขไอพีที่ใช้ในการบริการทางอินเตอร์เน็ต

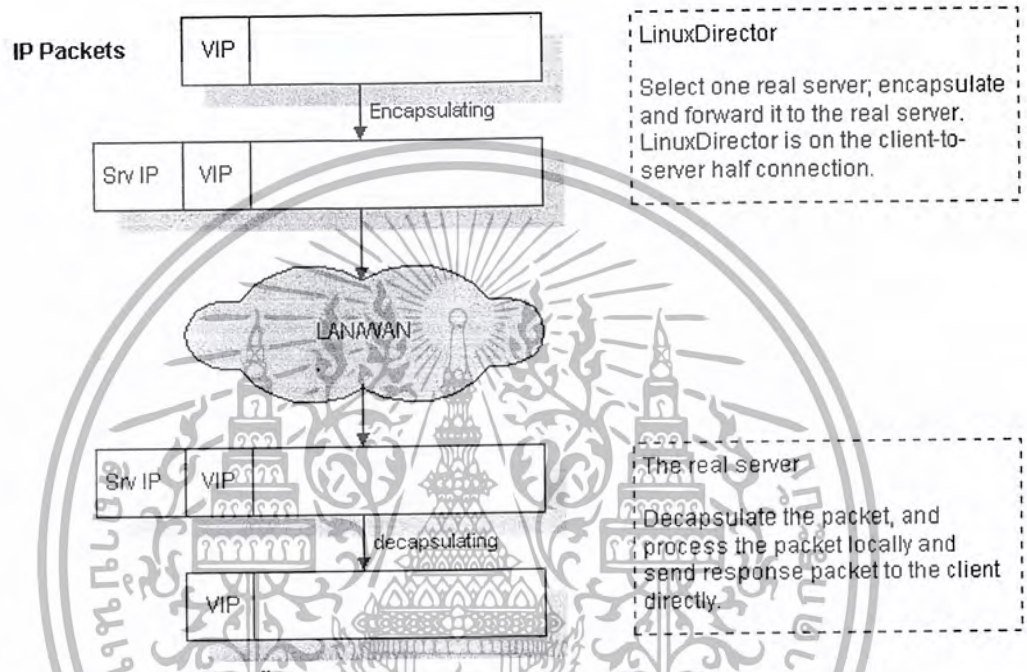


รูปที่ 2-6 สถาปัตยกรรมของระบบ Linux Virtual Server via IP Tunneling

ขั้นตอนการทำงานของ Linux Virtual Server via IP Tunneling นั้นจะเหมือนกันกับ Linux Virtual Server via NAT แต่จะมีข้อแตกต่างตรงที่ว่าวิธีการของ Linux Virtual Server via IP Tunneling นั้นเครื่องที่เป็นโหลดบาลานซ์เมื่อได้รับสัญญาณการร้องขอบริการทางอินเตอร์เน็ตก็จะทำการ

encapsulate กับ IP datagram ภายใน IP datagram อีกที่หนึ่งของ แพคเกต ที่มีเข้ามา และก็จะทำการส่ง แพคเกต นั้นไปให้กับเครื่องเซิร์ฟเวอร์ที่ได้ทำการเลือก เมื่อเครื่องเซิร์ฟเวอร์เครื่องที่ได้เลือกนั้นได้รับ แพคเกต ที่ผ่านการ encapsulate โดยเครื่องโหลดบาลานซ์ เครื่องเซิร์ฟเวอร์นั้นก็จะทำการ decapsulate กับ แพคเกต นั้นและพบว่าหมายเลขไอพีปลายทางที่จะส่งนั้นเป็นของ Virtual IP ซึ่งก็จะตรงกับ อินเตอร์เฟสแบบ tunnel ที่เครื่องเซิร์ฟเวอร์นั้นได้ทำการติดตั้งเป็น Virtual IP ส่งผลให้เครื่องเซิร์ฟเวอร์ เครื่องนั้นทำงานกับการร้องขอบริการและส่งผลลัพธ์ไปให้กับ ไคลเอนต์โดยตรง

ขั้นตอนการทำงานของวิธีการ Linux Virtual Server via IP Tunneling จะเป็นไปตามรูปที่ 2-7



รูปที่ 2-7 ขั้นตอนการทำงานของ Linux Virtual Server via IP Tunneling

จากรูปที่ 2-7 เครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ของระบบนี้สามารถที่จะมีหมายเลขไอพีจริงที่ สามารถใช้ได้ในอินเตอร์เน็ตที่จะอยู่ในเครือข่ายไหนก็ได้ ซึ่งจะทำให้ระบบนี้มีการกระจายไปตามพื้นที่ บริการต่างๆที่อยู่ห่างไกลกัน แต่เครื่องเซิร์ฟเวอร์ที่ใช้ในระบบนี้จำเป็นต้องสนับสนุนกับโพรโตคอลแบบ IP encapsulate โดยที่เครื่องเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์ทุกเครื่องจะทำการติดตั้งอุปกรณ์ อินเตอร์เฟสที่เป็นแบบ Tunnel ซึ่งจะทำให้เครื่องเซิร์ฟเวอร์ทุกเครื่องในคลัสเตอร์ทำการ decapsulate กับ แพคเกต ที่ถูก encapsulate มาโดยเครื่องที่โหลดบาลานซ์ และหมายเลขไอพีที่เป็น Virtual Address จะต้องทำการติดตั้งในอุปกรณ์อินเตอร์เฟสที่ไม่ทำการ ARP response

ในการทำงานนั้นเมื่อ แพคเกต ที่ถูก encapsulate มาถึงเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ ซึ่ง เครื่องเซิร์ฟเวอร์นี้ก็จะทำการ decapsulate และพบว่า แพคเกต นั้นส่งมาให้กับที่อยู่ปลายทางเป็น Virtual IP Address และมันก็จะพบว่าเป็นของที่เครื่องมันเอง ต่อจากนั้นมันก็จะทำงานกับ request นั้นและทำ การส่งผลลัพธ์ไปให้กับไคลเอนต์โดยตรง

ข้อเปรียบเทียบของวิธีการต่างๆ

ลักษณะเฉพาะของการส่งต่อแพคเกต ที่ใช้ใน Linux Virtual Server ทั้ง 3 วิธี สรุปในตารางที่ 2-1

	NAT	IP Tunneling	Direct Routing
Server	Any	Tunneling	Non-arp device
Server Network	Private	LAN/WAN	LAN
Server number	Low(10~20)	High	High
Server gateway	Load balancer	Own router	Own router

ตารางที่ 2-1 ข้อเปรียบเทียบของวิธีการต่างๆที่ใช้ใน Linux Virtual Server

2.5 Scheduling Algorithm

ในระบบ Linux Virtual Server จะมีวิธีการกระจายการร้องขอบริการไปให้กับเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์หรือที่เรียกว่า Scheduling Algorithm อยู่ทั้งหมด 4 วิธีคือ

1. Round – Robin Scheduling
2. Weighted Round – Robin Scheduling
3. Least – Connection Scheduling
4. Weighted Least – Connection Scheduling

จาก Scheduling Algorithm ข้างบนนั้นสองวิธีแรกจะไม่ได้คำนึงถึงข้อมูลภาระงานที่ทำงานอยู่ในแต่ละเครื่องของเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์ส่วนสองวิธีหลังนั้นจะทำการนับจำนวนการติดต่อที่กำลังทำงานอยู่ภายในแต่ละเครื่องเซิร์ฟเวอร์และจะทำการกระจายงานไปให้เครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์โดยคำนึงถึงจำนวนการติดต่อเป็นเกณฑ์ในการกระจายงาน

2.5.1 Round – Robin Scheduling

วิธีการทำงานของ Round – Robin Scheduling คือเมื่อเครื่องที่ทำหน้าที่เป็นโหลดบาลานซ์ได้รับการร้องขอบริการเข้ามาก็จะทำการกระจายการร้องขอบริการนั้นให้กับเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์แบบวนไปเรื่อยๆ โดยที่จะไม่ได้พิจารณาถึงจำนวนการติดต่อหรือเวลาตอบสนองการร้องขอบริการของเซิร์ฟเวอร์แต่ละเครื่องที่อยู่ภายในคลัสเตอร์

การทำงานของวิธีการนี้จะคล้ายกับการหลักการของ Round – Robin DNS ซึ่งหลักการของ Round – Robin DNS นั้นจะทำการจับคู่ชื่อโดเมนหนึ่งกับหมายเลขไอพีที่มากกว่าหนึ่ง ซึ่งหน่วยย่อยของการ Scheduling แบบนี้จะอยู่ในรูปของเครื่อง และการ cache ของ Round – Robin DNS นั้นจะทำให้ Round – Robin Scheduling ที่ใช้ไม่ได้ผลตามที่ต้องการ ซึ่งจะส่งผลให้เกิดการกระจายงานที่ไม่สมดุลในระหว่างเครื่องเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์ ซึ่งเมื่อเทียบกับ Round –Robin Scheduling ของ Linux Virtual Server แล้วหน่วยย่อยของการ Scheduling จะอยู่ในรูปของการติดต่อ ซึ่งมันจะมีประสิทธิภาพมากกว่าวิธีการของ Round – Robin DNS เนื่องจากหน่วยย่อยในการทำงานมีความละเอียดมากกว่า

2.5.2 Weighted Round – Robin Scheduling

วิธีการทำงานของ Weighted Round – Robin Scheduling คือจะทำงานคล้ายกับ Round – Robin Scheduling แต่เราสามารถที่จะทำการกระจายงานโดยคำนึงถึงประสิทธิภาพในการทำงานที่แตกต่างกันของเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์โดยวิธีการนี้จะทำการกำหนดค่าตัวเลขที่บ่งบอกถึงประสิทธิภาพในการทำงานของเซิร์ฟเวอร์แต่ละเครื่องที่อยู่ภายในคลัสเตอร์

ขั้นตอนการทำงานของ Weighted Round – Robin Scheduling จะเป็นไปดังนี้คือ ถ้าสมมติว่ามีเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์จำนวน n เครื่องคือ $S = \{S_0, S_1, ..., S_{n-1}\}$ โดยที่ดัชนี i จะแทนเครื่องเซิร์ฟเวอร์ที่เพิ่งได้รับการเลือกในการกระจายงานโดยเครื่องโหลดบาลานซ์ และตัวแปร cw จะแทนค่าถ่วงน้ำหนักปัจจุบันในขณะนั้น โดยที่ตัวแปร i และ cw จะมีค่าเริ่มต้นเป็นศูนย์ ถ้าทุก $W(S_i)$ มีค่าเท่ากับศูนย์ แสดงว่าไม่มีเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ทำงานได้ เพราะฉะนั้นทุกการติดต่อที่เข้ามายัง Virtual Server จะไม่ทำงานซึ่ง algorithm จะเป็นไปตามข้างล่างนี้

```
While(1) {
    if (i==0){
        cw = cw -1;
        if (cw <= 0){
            set cw the maximum weight of S;
            if ( cw == 0) return NULL;
            }
        }
    else i = ( i+1) mod n;
    if ( W(Si) >= cw) return Si;
    }
```

วิธีการ Scheduling แบบนี้ทุกเซิร์ฟเวอร์ที่มีค่าถ่วงน้ำหนักมากจะทำการรับสัญญาณร้องขอก่อนและที่การติดต่อที่มากกว่าเครื่องเซิร์ฟเวอร์ที่มีค่าถ่วงน้ำหนักน้อยกว่า และเครื่องเซิร์ฟเวอร์ที่มีค่าถ่วงน้ำหนักเท่ากันก็จะได้รับการติดต่อของสัญญาณการร้องขอที่มีจำนวนเท่ากันด้วย เช่น ถ้ามีเซิร์ฟเวอร์ A, B และ C ซึ่งมีค่าถ่วงน้ำหนักเป็น 4, 3 และ 2 ตามลำดับ แล้วการ Scheduling ที่ดีในช่วงเวลาหนึ่งจะเป็นดังนี้คือ ABCABCABA

วิธีการแบบ Weighted Round – Robin Scheduling นั้นจะไม่ได้คำนึงถึงจำนวนการติดต่อของแต่ละเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์ ซึ่งทำให้มี overhead น้อย และส่งผลให้ระบบคลัสเตอร์สามารถมีจำนวนเซิร์ฟเวอร์ภายในคลัสเตอร์ได้มาก แต่อย่างไรก็ตามวิธีการนี้อาจนำไปสู่การกระจายงานอย่างไม่สมดุลภายในระบบคลัสเตอร์ได้ถ้าภาระของการร้องขอบริการมีความแตกต่างกันมาก กล่าวคือภาระการร้องขอบริการจำนวนมากที่มีการทำงานมากอาจจะถูกส่งไปให้กับเซิร์ฟเวอร์เครื่องหนึ่งที่อยู่ภายในคลัสเตอร์มากกว่าเครื่องอื่นได้

2.5.3 Least – Connection Scheduling

วิธีการทำงานของ Least – Connection Scheduling คือจะทำการกระจายงานไปให้กับเซิร์ฟเวอร์ที่มีจำนวนการติดต่ออยู่กับการร้องขอบริการที่มีอยู่น้อยที่สุด โดยวิธีการนี้จะต้องทำการนับการติดต่อที่ยังทำงานอยู่ (ภายในเซิร์ฟเวอร์แต่ละเครื่องที่อยู่ภายในคลัสเตอร์ ซึ่งวิธีการแบบนี้จะเหมาะสำหรับระบบคลัสเตอร์ที่เซิร์ฟเวอร์แต่ละเครื่องมีความสามารถในการทำงานที่ใกล้เคียงกัน และเหมาะสำหรับภาระการร้องขอบริการที่มีการทำงานแตกต่างกันมาก เพราะว่าภาระการร้องขอบริการที่มีการทำงานมากจะไม่มีโอกาสได้ส่งไปยังเซิร์ฟเวอร์เครื่องหนึ่งเครื่องใดที่อยู่ภายในคลัสเตอร์เพียงเครื่องเดียว

การกระจายงานอย่างนี้จะทำงานได้ไม่ดีเมื่อแต่ละเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์มีประสิทธิภาพในการทำงานที่แตกต่างกัน สาเหตุก็เนื่องมาจากสถานะ TIME_WAIT ของโพรโตคอล TCP ซึ่งโดยทั่วไปจะมีค่าสองนาที ซึ่งในเวลาสองนาทีนี้เวปไซต์ที่เป็นที่นิยมสามารถรับการร้องขอบริการที่มีเข้ามาได้มากกว่าจำนวนหนึ่งพันการร้องขอบริการ ตัวอย่างเช่น เมื่อเซิร์ฟเวอร์เครื่องหนึ่งมีประสิทธิภาพในการทำงานเป็นสองเท่าของอีกเครื่องหนึ่งที่อยู่ภายในคลัสเตอร์เดียวกัน เครื่องที่ทำงานเร็วกว่ากำลังทำงานกับการร้องขอบริการที่มีจำนวนเป็นพันการร้องขอบริการ และได้เก็บงานการร้องขอบริการนั้นไว้ในสถานะ TIME_WAIT ของโพร โตคอล TCP ส่วนเครื่องที่ทำงานช้ากว่าได้ทำงานกับการร้องขอบริการที่มีมาเรื่อยๆ และในที่สุดก็ได้ทำงานกับการร้องขอบริการที่มีเข้ามาเป็นจำนวนหนึ่งพันการร้องขอบริการเสร็จ และยังคงรับสัญญาณร้องขอที่มีเข้ามาได้เรื่อยๆด้วยเหตุนี้การทำงานของ Least – Connection Scheduling จะทำการกระจายงานไปให้เซิร์ฟเวอร์ต่างๆภายในคลัสเตอร์ได้อย่างไม่สมดุลเมื่อเซิร์ฟเวอร์แต่ละเครื่องภายในคลัสเตอร์มีประสิทธิภาพในการทำงานที่แตกต่างกัน

2.5.4 Weighted Least – Connection Scheduling

วิธีการทำงานของ Weighted Least – Connection Scheduling คือการทำงานคล้ายกับ Least – Connection Scheduling แต่เราสามารถที่จะทำการกระจายงานโดยคำนึงถึงประสิทธิภาพในการทำงานที่แตกต่างกันของเซิร์ฟเวอร์ต่างๆที่อยู่ภายในคลัสเตอร์ โดยวิธีการนี้จะทำการกำหนดค่าตัวเลขที่บ่งบอกถึงประสิทธิภาพในการทำงานของเซิร์ฟเวอร์ที่มีค่าถ่วงน้ำหนักสูงกว่าเมื่อเทียบกับเครื่องอื่นจะทำการรับภาระการร้องขอบริการมากเมื่อเทียบกับเครื่องอื่นด้วยซึ่งผู้ดูแลระบบสามารถทำการกำหนดค่าถ่วงน้ำหนักนี้ให้กับแต่ละเซิร์ฟเวอร์ที่อยู่ภายในคลัสเตอร์ ซึ่งการกระจายงานจะทำโดยคำนึงถึงจำนวนการติดต่อของเซิร์ฟเวอร์แต่ละเครื่องและค่าถ่วงน้ำหนักที่กำหนดให้แต่ละเซิร์ฟเวอร์

การทำงานของ Weighted Least – Connection Scheduling จะเป็นไปดังนี้ สมมติว่ามีเซิร์ฟเวอร์จำนวน n เครื่อง แต่ละเซิร์ฟเวอร์ i มีค่าถ่วงน้ำหนักเป็น W_i (i = 1, … , n) และจำนวนการติดต่อของแต่ละเซิร์ฟเวอร์ i เป็น C_i(i = 1, …, n) โดยที่ผลรวมของการติดต่อทั้งหมด C_i(i = 1, …, n) เป็น T แล้วการติดต่อครั้งต่อไปจะถูกส่งไปให้กับเซิร์ฟเวอร์ j ซึ่ง

$$(C_j / T) / W_j = \text{MIN}\{(C_i / T) / W_i\} \quad (i = 1, \ldots, n)$$

d เพราะว่า T เป็นค่าคงที่ จึงเขียนให้อยู่ในรูปที่ง่ายได้เป็น

$$C_j / W_j = \text{MIN} \{ C_i / W_i \} \ (i = 1, \ldots, n)$$

เนื่องมาจากว่าใน Kernel ของลีนุกซ์จะไม่มีจำนวนแบบ float การเปรียบเทียบค่าของ $C_j / W_j > C_i / W_i$ จะเปลี่ยนเป็น $C_j * W_i > C_i / W_j$ เพราะว่าค่าถ่วงน้ำหนักจะมีค่าเริ่มต้นเป็นหนึ่งและจะไม่มีทางมีค่าเป็นศูนย์

2.6 การติดตั้ง LVS

Linux บางค่ายนั้นได้ทำการรวม LVS ไว้ใน Kernel อยู่แล้ว อย่างเช่น SuSE ในกรณีนี้การติดตั้งจะทำได้ง่ายเพียงแค่ติดตั้งแพคเกจของ ipvsadm ลงไปเท่านั้น สำหรับบางค่ายที่ไม่ได้ทำการติดตั้ง LVS มาให้แล้วนั้นก็ต้องทำการติดตั้งลงไปเอง โดยมีขั้นตอนของการติดตั้งดังนี้

1. Unpack Kernel

จากตัวอย่างนี้เราจะใช้ Kernel 2.4.20 ซึ่งเวอร์ชั่นใหม่ๆ นั้นสามารถหาได้จาก www.kernel.org เมื่อเราได้ มาแล้วให้ทำการ Unpack ไฟล์ที่ได้มาลงไปในไดเรคทอรี linux-2.4.20

```
tar -jxvf linux-2.4.20.tar.bz2
```

รูปที่ 2-8 วิธีการ Unpack Kernel

2. Unpack LVS

จากตัวอย่างนี้เราจะใช้เวอร์ชั่น 1.0.9 ซึ่งเวอร์ชั่นใหม่ๆ นั้นสามารถหาได้จาก www.linuxvirtualserver.org ให้เราทำการ Unpack ไฟล์ที่ได้มาลงไปในไดเรคทอรี่ ipvs-1.0.9

```
tar -zxvf ipvs-1.0.9.tar.gz
```

รูปที่ 2-9 วิธีการ Unpack LVS

3. ทำการ Patch LVS ลงไปใน Kernel

มีด้วยกัน 2 ไฟล์ที่เราจะต้อง Patch ลงไป ได้แก่ linuxkernel_ksyms_c.diff และ linuxnet_netsyms_c.diff

```
cd linux-2.4.20/
patch -pq < ../ipvs-1.0.9/linuxkernel_ksyms_c.diff
patch -pq < ../ipvs-1.0.9/linuxnet_netcyms_c.diff
```

รูปที่ 2-10 วิธีการ Patch LVS ลงไปใน Kernel

4. Configure Kernel

มีด้วยกันหลายวิธีที่จะทำการคอนฟิกนี้ เช่น make menuconfig , make xconfig และ make config แต่ต้องระลึกไว้เสมอว่าวิธีการที่เราใช้นี้จะต้องทำการคอมไพล์ให้สนับสนุน netfilter ด้วย ซึ่งออปชั่น เหล่านี้เป็นสิ่งจำเป็น

```
Networking options  --->
  Network packet filtering (replaces ipchains)
  <m> IP: tunnelling
  IP: Netfilter Configuration --->
     <m> Connection tracking (required for masq/NAT)
       <m> FTP protocol support
     <m> IP tables support (required for filtering/masq/NAT)
       <m> Packet filtering
         <m> REJECT target support
       <m> Full NAT
         <m> MASQUERADE target support
         <m> REDIRECT target support
         <m> NAT of local connections (READ HELP) (NEW)
       <m> Packet mangling
         <m> MARK target support
       <m> LOG target support
```

รูปที่ 2-11 ออปชั่นที่ต้องการในการคอนฟิก Kernel

5. Build and Install Kernel

```
make dep
make bzImage modules
make install modules_install
```

รูปที่ 2-12 ขั้นตอนการสร้างและการติดตั้ง Kernel

6. Update boot loader

หากว่าเราใช้ grub เป็น boot loader ให้เราแก้ไฟล์ /etc/grub.conf เช่น

```
title 2.4.20 LVS
        root (hd0,0)
        kernel /vmlinuz-2.4.20 ro root=/dev/hda3
```

รูปที่ 2-13 วิธีการ Update boot loader กรณีใช้ grub

หากว่าเราใช้ lilo เป็น boot loader ให้เราแก้ไฟล์ /etc/lilo.conf เช่น

```
image=/boot/vmlinuz-2.4.20
        label=2.4.20-lvs
        read-only
        root=/dev/hda2
```

รูปที่ 2-14 วิธีการ Update boot loader กรณีใช้ lilo

7. Reboot the system

เมื่อเราทำการ reboot ระบบแล้วให้สังเกต boot loader ที่เราได้ทำการแก้ไขแล้วด้วย เพื่อเป็นการแน่ในว่า เราได้ boot ระบบขึ้นมาด้วย Kernel ที่เราต้องการ

8. Build and Install LVS

คอมมานด์ที่เราทำการสร้าง LVS นั้น ควรที่จะทำการรันจากไดเรคทอรี ipvs-1.0.9/ipvs/ การสร้างและการติดตั้งนั้นสามารถทำได้ดังตัวอย่างนี้

```
make KERNELSOURCE=/kernel/source/linux-2.4.20 all
make KERNELSOURCE=/kernel/source/linux-2.4.20 modules_install
```

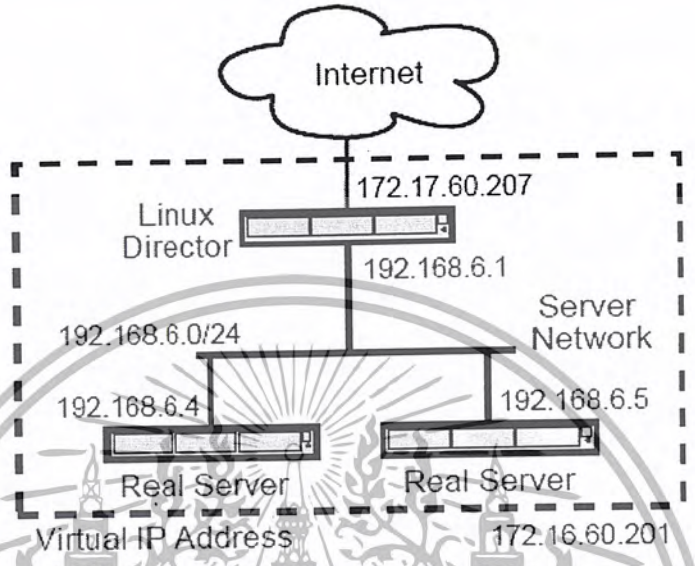
รูปที่ 2-15 วิธีการสร้างและการติดตั้ง LVS

9. Build and Install Ipvsadm

Ipvsadm นั้นเป็นเครื่องมือที่ใช้ในการคอนฟิก LVS เราสามารถหาซอสโค้ดได้ที่ไดเรคทอรี /ipvs-1.0.9/ipvs/ipvsadm การสร้างและการติดตั้ง

2.7 การติดตั้ง LVS แบบ NAT

เป็นวิธีการที่ง่ายที่สุดในการที่จะคอนฟิก LVS โดยแพคเกตจาก Linux Director จะส่งให้กับ Real Server ซึ่งปลายทางของผู้รับจะถูกเขียนใหม่เพื่อที่จะทำการส่งไปให้ถูก Real Server ส่วนการส่งแพคเกตกลับมาจาก Real Server นั้นจะส่งกลับมาทางเดิมให้กับ Linux Director อีกครั้ง



รูปที่ 2-16 ตัวอย่าง LVS NAT

Linux Director

- Enable IP forwarding สามารถทำได้โดยการเพิ่มบรรทัดด้านล่างนี้ลงไปในไฟล์ /etc/sysctl.conf แล้วรันคำสั่ง sysctl -p

```
net.ipv4.ip_forward = 1
```

รูปที่ 2-17 คำสั่งที่เพิ่มเข้าไปในไฟล์ /etc/sysctl.conf

- Bring up 172.17.60.201 บน eth0:0 สามารถทำได้โดย

```
ifconfig eth0:0 172.17.60.201 netmask 255.255.0.0 broadcast 172.17.255.255
```

รูปที่ 2-18 คำสั่งการ Bring up IP address

- คอนฟิก LVS

```
ipvsadm -A -t 172.17.60.201:80
ipvsadm -a -t 172.17.60.201:80 -r 192.168.6.4:80 -m
ipvsadm -a -t 172.17.60.201:80 -r 192.168.6.5:80 -m
```

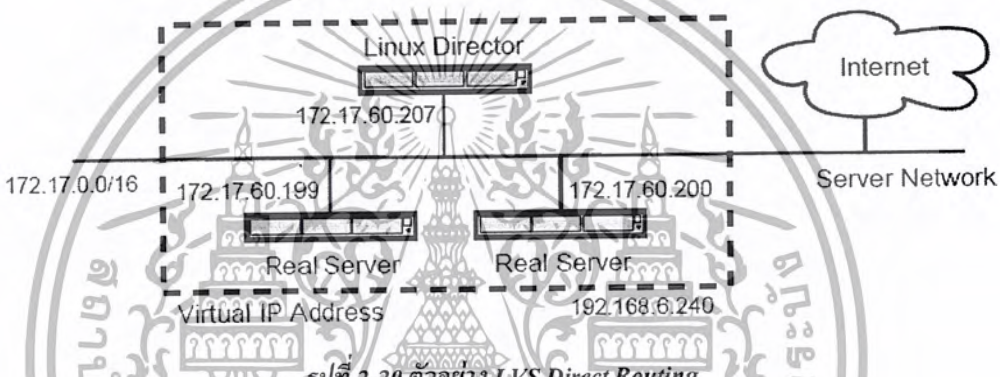
รูปที่ 2-19 การคอนฟิก LVS โดยอ้างอิง IP address จากรูป

Real Server

- ทำให้แน่ในว่ามีการส่งแพคเกตผ่านกลับมาทาง Linux Director สามารถทำได้โดยการเซตเกตเวให้วิ่งไปที่ 192.168.6.1
- ทำให้แน่ใจว่ามีการเปิดการบริการโดยการตรวจสอบจากพอตที่เปิดไว้ เพื่อให้ผู้ใช้บริการของระบบเข้ามาใช้บริการได้

2.8 การติดตั้ง LVS แบบ Direct Routing

การทำงานในแบบ Direct Routing สามารถทำได้โดยการส่งผ่านแพคเกตแบบไม่เปลี่ยนแปลงไปที่ MAC address ของ Real Server ดังนั้นตัว Real Server จึงไม่จำเป็นที่จะต้องทำการส่งแพคเกตกลับไปยังตัว Linux Director ดังนั้นการเกิดปัญหาคอขวดที่ Linux Director จึงลดลง



รูปที่ 2-20 ตัวอย่าง LVS Direct Routing

Linux Director

- Enable IP forwarding สามารถทำได้โดยการเพิ่มบรรทัดด้านล่างนี้ลงไปในไฟล์ /etc/sysctl.conf แล้วรันคำสั่ง sysctl –p

```
net.ipv4.ip_forward = 1
```

รูปที่ 2-21 คำสั่งที่เพิ่มเข้าไปในไฟล์ /etc/sysctl.conf

- Bring up 172.17.60.201 บน eth0:0 สามารถทำได้โดย

```
ifconfig eth0:0 172.17.60.201 netmask 255.255.0.0 broadcast 172.17.255.255
```

รูปที่ 2-22 คำสั่งการ Bring up IP address

- คอนฟิก LVS

```
ipvsadm -A -t 172.17.60.201:80
ipvsadm -a -t 172.17.60.201:80 -r 172.17.60.199:80 -g
ipvsadm -a -t 172.17.60.201:80 -r 172.17.60.200:80 -g
```

รูปที่ 2-23 การคอนฟิก LVS โดยอ้างอิง IP address จากรูป

- Real Server นั้นสามารถที่จะส่งกลับแพคเกตไปให้กับผู้ที่เข้ามาขอใช้บริการได้เลย ดังนั้น ตัว Linux Director จึงไม่จำเป็นที่จะต้องทำหน้าที่เป็นเกตเวให้กับ Real Server

Real Server

- ทำให้แน่ใจว่าไม่มีการส่งกลับของแพคเกตไปที่ Linux Director
- ทำให้แน่ใจว่ามีการเปิดการบริการ โดยการตรวจสอบจากพอตที่เปิดไว้ เพื่อให้ผู้ใช้บริการของระบบเข้ามาใช้บริการได้
- Bring up 172.17.60.201 บน loopback interface สามารถทำได้โดย

```
ifconfig lo:0 172.17.60.201 netmask 255.255.255.255
```

รูปที่ 2-24 คำสั่งการ Bring up loopback interface

- ทำการซ่อน loopback บนตัว Real Server จำเป็นที่จะต้องมีการซ่อน loopback interface เพื่อเป็นการป้องกันจากการตอบสนองต่อ ARP รีเควส โดยเราสามารถทำได้โดยการเพิ่มบรรทัดเหล่านี้เข้าไปในไฟล์ /etc/sysctl.conf และรันคำสั่ง sysctl -p

```
# Enable configuration of hidden devices
net.ipv4.conf.all.hidden = 1
# Make the loopback interface hidden
net.ipv4.conf.lo.hidden = 1
```

รูปที่ 2-25 คำสั่งที่ถูกเพิ่มเข้าไปในไฟล์ /etc/sysctl.conf

2.9 การติดตั้ง LVS แบบ Tunneling

การทำงานของ LVS Tunneling นั้นจะเหมือนกับการทำงานในแบบ Direct Routing แต่ส่วนที่แตกต่างกันอย่างเห็นได้ชัดนั้นคือการส่งต่อแพคเกตไปที่ Real Server นั้นจะใช้หลักการ IP encapsulate ใน IP โดยข้อได้เปรียบจากหลักการนี้ก็คือตัว Real Server นั้นสามารถอยู่คนละเครือข่ายกับ Linux Director ได้

Linux Director

- Enable IP forwarding สามารถทำได้โดยการเพิ่มบรรทัดด้านล่างนี้ลงไปในไฟล์ /etc/sysctl.conf แล้วรันคำสั่ง sysctl –p

```
net.ipv4.ip_forward = 1
```

รูปที่ 2-26 คำสั่งที่เพิ่มเข้าไปในไฟล์ /etc/sysctl.conf

- Bring up 172.17.60.201 บน eth0:0 สามารถทำได้โดย

```
ifconfig lo:0 172.17.60.201 netmask 255.255.255.255
```

รูปที่ 2-27 คำสั่งการ Bring up IP address

- คอนฟิก LVS

```
ipvsadm -A -t 172.17.60.201:80
ipvsadm -a -t 172.17.60.201:80 -r 172.17.60.199:80 -i
ipvsadm -a -t 172.17.60.201:80 -r 172.17.60.200:80 -i
```

รูปที่ 2-28 การคอนฟิก LVS โดยอ้างอิง IP address จากรูป

Real Server

- ทำให้แน่ใจว่าไม่มีการส่งกลับของแพคเกตไปที่ Linux Director
- ทำให้แน่ใจว่ามีการเปิดการบริการ โดยการตรวจสอบจากพอตที่เปิดไว้ เพื่อให้ผู้ใช้บริการของระบบเข้ามาใช้บริการได้
- Bring up 172.17.60.201 บน tunl0 สามารถทำได้โดย

```
ifconfig tunl0 172.17.60.201 netmask 255.255.255.255
```

รูปที่ 2-29 คำสั่งการ Bring up IP address

- ทำการ Enable การส่งแพคเกตและซ่อน loopback โดยการแก้ไขไฟล์ /etc/sysctl.conf โดยการเพิ่มคำสั่งด้านล่างนี้ลงไปแล้วทำการรันคำสั่ง sysctl –p

```
net.ipv4.ip_forward = 1
# Enable configuration of hidden devices
net.ipv4.conf.all.hidden = 1
# Make the tunl0 interface hidden
net.ipv4.conf.tunl0.hidden = 1
```

รูปที่ 2-30 การ Enable และการ ซ่อน loopback interface

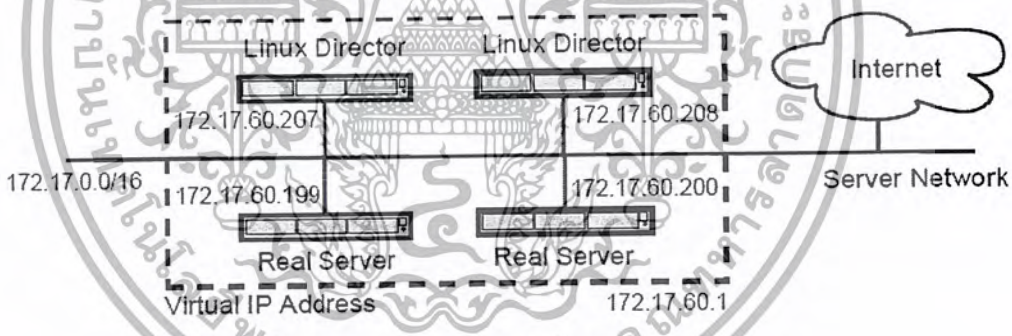
บทที่ 3

High Availability

LVS นั้นเป็นเป็นวิธีการที่มีประสิทธิภาพในการทำเครือข่ายที่ให้การบริการแบบโหลดบาลานซ์ โดยปกติแล้ว Server ที่ทำการให้บริการนั้นจะต้องสามารถทำงานได้นานตราบเท่าที่ผู้ใช้บริการต้องการ ดังนั้นการที่เรามี Server เพียงตัวเดียวแล้วจะต้องทำงานให้ได้ตราบเท่าที่ผู้ใช้บริการต้องการนั้นจึงเป็นเรื่องที่เป็นไปได้ยาก การที่เรามี Server ให้บริการหลายตัวในระบบจึงเป็นการเพิ่มโอกาสในการที่เราจะให้บริการให้นานตราบเท่าที่ผู้ใช้บริการต้องการได้มากกว่าการที่เรามี Server ให้บริการเพียงตัวเดียว

3.1 Heartbeat

Heartbeat นั้นจะใช้มอนิเตอร์คู่ของ Linux Director เพื่อทำให้แน่ใจว่าไม่ว่าเวลาใดก็ตามจะต้องมีตัวใดตัวหนึ่งจากสองตัวนี้เป็นผู้ถือครอง VIP ตัว Heartbeat สามารถทำงานได้โดยการส่ง heartbeat message ในช่วงเวลาที่กำหนดไว้ ถ้าหากไม่ได้รับ heartbeat message ภายในช่วงเวลาที่คาดการไว้แล้ว Host ก็จะสามารถสรุปได้ว่ามีการล้มเหลวเกิดขึ้นกับ Server อีกตัว ดังนั้นการเทคโอเวอร์รีซอสของระบบก็จะเกิดขึ้น (ในที่นี้จะหมายถึง IP address) เราสามารถหา package ของ heartbeat ได้จาก www.linux-ha.org สำหรับตัวอย่างเครือข่ายที่ทำการติดตั้ง Heartbeat นั้นแสดงไว้ดังรูปที่ 3-1



รูปที่ 3-1 ตัวอย่างการเชื่อมต่อที่มี Heartbeat

การคอนฟิก Heartbeat นั้นสามารถทำได้โดยแก้ไขไฟล์คอนฟิก 3 ไฟล์ซึ่งจะอยู่ที่ /etc/ha.d/ ดังนี้

- ha.cf : ไฟล์นี้จะเป็นการเก็บค่าพารามิเตอร์หลักสำหรับ Heartbeat อย่างเช่นลักษณะของการเชื่อมต่อ , วิธีการส่ง Message และที่ที่เราจะทำการเก็บ log file และจะต้องระลึกไว้เสมอว่า ชื่อของ node ที่อยู่ในไฟล์นี้จะเป็นชื่อของเครื่องของ Director ที่อยู่ในระบบคลัสเตอร์และจะต้องเป็นชื่อเดียวกับที่คำสั่ง uname –n รีเทิร์นออกมาด้วย
- haresources : เป็นกลุ่มของ resource ซึ่งจะถูกจัดการโดย heartbeat

- authkeys : เป็นวิธีการทางด้านการรักษาความปลอดภัยสำหรับการติดต่อระหว่าง heartbeat ไฟล์นี้จะต้องเซตให้เป็นโหมด 600

ตัวอย่างของไฟล์คอนฟิกในการออกแบบระบบ สามารถอ่านได้จากภาคผนวก ก.

ข้อควรจำ : การคอนฟิก Linux Director แต่ละตัวนั้น จะต้องคอนฟิกให้เหมือนกันด้วย จะต่างกันก็แต่เพียงชื่อของ node ในไฟล์ haresources เท่านั้น

3.1.1 IP address take-over

ถ้าหาก node ที่ทำหน้าที่ให้บริการอยู่ในขณะนั้นเกิดการล้มเหลวขึ้นมา อีก node ที่เหลือจะทำวิธีการทาง ARP เพื่อให้ได้มาซึ่ง IP address ที่ node แรกถือครองอยู่แล้วนั้น เราจะเรียกวิธีการนี้ว่า IP address take-over

3.2 Ldirectord

เมื่อ Heartbeat ใช้ทำการมอนิเตอร์คู่ของ Linux Director แล้ว เราจะใช้ Ldirectord ในการมอนิเตอร์ตัว Real Server และเพื่อทำการจัดการกับ LVS Kernel table ด้วย ตัว Ldirector และ Heartbeat นั้นส่วนมากจะใช้ร่วมกันในการสร้างระบบ High availability LVS cluster

Ldirector จะทำการตรวจสอบการให้บริการบน Real Server โดยทำการติดต่อไปยัง Real Server เหล่านั้น จะมีลักษณะคล้ายกับว่าเป็นผู้ขอใช้บริการเอง และ จะรอรับผลการตอบรับจาก Real Server และตรวจสอบค่าที่คาดว่าจะได้รับกลับมาในรูปแบบของ String โดยที่การตรวจสอบนั้นสามารถทำได้บนโพรโตคอล HTTP, HTTPS, FTP, IMAP, POP, SMTP, LDAP และ NNTP ในการตรวจสอบ String ที่คาดว่าจะได้รับกลับมานั้นเราสามารถทำการคอนฟิกได้ว่าเราคาดว่าจะได้รับ String ใดกลับมา

ตัวอย่างของไฟล์คอนฟิก สามารถอ่านได้จากภาคผนวก ก.

3.3 Ultra monkey

Ultra monkey คือโปรเจค (Project) ที่จะทำการสร้างการให้บริการแบบ Load balance และ High available บนระบบเครือข่าย โดยจะทำงานบนระบบปฏิบัติการที่เป็นลินุกซ์ ซึ่งจะเน้นหลักไปทางการขยายขนาดและการทำงานแบบไม่ล้มเหลวของการให้บริการแบบ WEB แต่ด้วยเทคโนโลยีนี้เราก็สามารถที่จะทำการขยายขนาดของการให้บริการไปสู่การบริการลักษณะอื่นๆ ได้เช่น จดหมายอิเล็กทรอนิกส์ หรือ ระบบขนส่งไฟล์ผ่านเครือข่าย

ระบบโดยรวม

- สามารถเพิ่มจำนวน IP ที่ให้บริการในส่วนของ Real Server ได้
- ส่วนของ High Availability จะทำโดยการใช้โพรโตคอล Heartbeat
- การตรวจสอบการให้บริการต่างๆ จะทำโดยการใช้ Ldirectord

โดยที่เราสามารถหาดาวโหลด Ultra monkey ได้จาก www.ultramonkey.org จะมีทั้งแบบ RPM และแบบที่เป็น Source Code ให้เราเลือกใช้งาน

จากขั้นตอนการใช้งานและการติดตั้ง LVS ที่ผ่านมาในบนที่ 2 นั้นเราจะเห็นได้ว่ามีความยุ่งยากในการติดตั้งพอสมควร ดังนั้นหากเราใช้ Ultra monkey ที่เป็นการรวบรวมขั้นตอนการคอนฟิกระบบต่างๆ ที่จำเป็น ไว้ในรูปแบบของ RPM เรียบร้อยแล้ว ก็จะทำให้การติดตั้ง LVS ของเราสามารถทำได้สะดวกรวดเร็วขึ้นอีกมาก

3.3.1 ขั้นตอนการติดตั้ง Ultra monkey

ในการทำโปรเจคนี้เราได้ใช้ระบบปฏิบัติการเป็น Red Hat 8.0 ดังนั้น คำสั่งต่างๆ ที่ใช้ในการติดตั้งจึงอ้างอิงจากระบบปฏิบัติการนี้ สำหรับการติดตั้งในลินุกซ์ตัวอื่นๆ นั้นสามารถอ่านวิธีติดตั้งได้จาก www.ultramonkey.org

- **Upgrade Kernel**

  จากการทำงานของ Linux Virtual Server ที่ไม่ได้ถูกรวมเข้ามาใน Kernel ของ Red Hat 8.0 ทำให้เราต้องทำการปรับแต่ง Kernel ให้เหมาะสมก่อนการติดตั้ง เพื่อให้เป็นการง่ายต่อการติดตั้ง ควรจะเป็น Kernel ใหม่ๆ ที่ยังไม่มีการปรับปรุงใดๆ เลย จากตัวอย่าง จะเป็นการอ้างอิงจาก Kernel เวอร์ชั่น 2.4.20-19 ดังนั้นหากเราติดตั้ง Red Hat 8.0 ใหม่ๆ จะทำให้ได้ Kernel ที่ไม่ตรงกับไฟล์ของ Ultra monkey ที่ต้องการ (ทำให้เราไม่สามารถ patch เข้าไปได้) เราต้องทำการติดตั้ง Kernel ตัวที่ Ultra monkey ต้องการเสียก่อน เมื่อเราทำการติดตั้งเรียบร้อยแล้วให้ทำการ patch โดยมีวิธีดังนี้

  สำหรับคอมพิวเตอร์ที่มีสถาปัตยกรรมในแบบใดนั้นให้ใช้ RPM ที่ใช้เฉพาะสถาปัตยกรรมของตนเท่านั้น โดยให้ทำการ patch ทุกไฟล์ที่ลงท้ายด้วยสถาปัตกรรมของตน ดังตัวอย่าง

  สถาปัตยกรรม i386: rpm -Fhv kernel*2.4.20-19.8.um.1.i386.rpm

  สถาปัตยกรรม i586: rpm -Fhv kernel*2.4.20-19.8.um.1.i586.rpm

  สถาปัตยกรรม i686: rpm -Fhv kernel*2.4.20-19.8.um.1.i686.rpm

  สถาปัตยกรรม Athlon: rpm -Fhv kernel*2.4.20-19.8.um.1.athlon.rpm

หมายเหตุ : * หมายถึงทุกๆ ไฟล์ที่ขึ้นต้นด้วย kernel และลงท้ายด้วย 2.4.20-19.8.um1

- **Update Boot Loader**

  โดยปกติแล้วถ้าหากเราใช้ grub เป็น Boot Loader แล้วตัว Boot Loader ของเราจะถูกแก้ไขโดยอัตโนมัติ โดยเราสามารถตรวจสอบได้จาก /etc/grub.conf

  และถ้าหากเราใช้ lilo เป็น Boot Loader เราจำเป็นที่จะต้องแก้เอง โดยสามารถทำได้โดยแก้ไขไฟล์ /etc/lilo.conf

- **ติดตั้งแพคเกจอื่นๆ ที่เหลือ**

  ตัว Ultra monkey นั้นมาพร้อมกับแพคเกจอื่นๆ ที่เพิ่มเข้ามาอีกตัวอย่างเช่น Heartbeat และ Ldirectord โดยแพคเกจเหล่านี้ให้เราทำการติดตั้งไปเฉพาะบนเครื่องที่ทำหน้าที่เป็น Linux Director เท่านั้น จากการติดตั้งแพคเกจบางครั้งแพคเกจที่ใช้ในการติดตั้งอาจต้องการแพคเกจอื่นๆ ด้วย (package dependency) ให้เราทำการติดตั้งแพคเกจทั้งหมดตามลำดับต่อไปนี้เพื่อป้องกันการเกิดเหตุการณ์ดังกล่าว

  heartbeat-pils-1.0.3-1.rh.8.0.um.1.i386.rpm
  heartbeat-stonith-1.0.3-1.rh.8.0.um.1.i386.rpm
  heartbeat-1.0.3-1.rh.8.0.um.1.i386.rpm
  perl-Net-SSLeay-1.22-1.rh.8.0.um.1.i386.rpm
  perl-Mail-IMAPClient-2.2.7-1.rh.8.0.um.1.noarch.rpm
  perl-IO-Socket-SSL-0.92-1.rh.8.0.um.1.noarch.rpm
  perl-Convert-ASN1-0.16-2.rh.8.0.um.1.noarch.rpm
  perl-Authen-SASL-2.03-1.rh.8.0.um.1.noarch.rpm
  perl-XML-NamespaceSupport-1.08-1.rh.8.0.um.1.noarch.rpm
  perl-XML-SAX-0.12-1.rh.8.0.um.1.noarch.rpm
  perl-ldap-0.2701-1.rh.8.0.um.1.noarch.rpm
  perl-Parse-RecDescent-1.80-1.rh.8.0.um.1.noarch.rpm
  heartbeat-ldirectord-1.0.3-1.rh.8.0.um.1.i386.rpm
  libnet-1.1.0-1.rh.8.0.um.1.i386.rpm
  ipvsadm-1.21-1.rh.8.0.um.1.i386.rpm

  การติดตั้งทุกๆ แพคเกจให้ใช้คำสั่ง rpm –Uvh แล้วตามด้วยชื่อของแพคเกจนั้นๆ

หมายเหตุ : ก่อนการติดตั้งแพคเกจที่กล่าวมานั้น ให้ตรวจสอบก่อนว่าเราได้ติดตั้งแพคเกจต่อไปนี้เรียบร้อยแล้ว

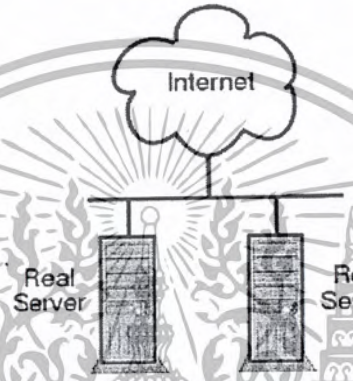
perl-Digest-HMAC และ perl-Digest-SHA1

- ทำการรีบูทระบบขึ้นมาใหม่และเลือก Kernel ที่เราต้องการขึ้นมาใช้งาน

บทที่ 4
การวิเคราะห์และการออกแบบ

ระบบที่สร้างขึ้นมานั้น สามารถทำงานได้ทั้งแบบที่เป็น High Availability, Load balance และทำงานทั้งสองแบบข้างต้นในเวลาเดียวกันได้ด้วย ดังนั้นการออกแบบจึงแบ่งตามประเภทของงานที่เราต้องการ

4.1 การออกแบบให้ทำงานในแบบ High Availability



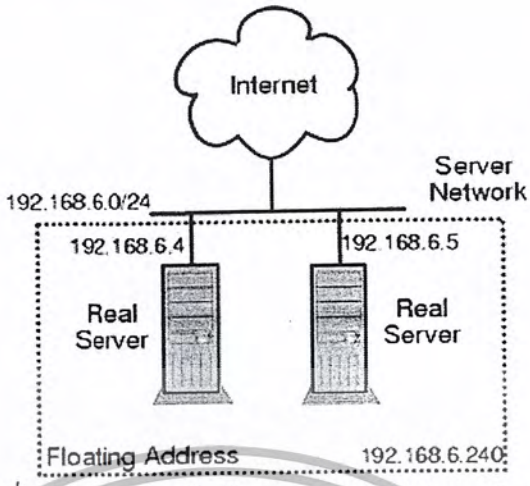
รูปที่ 4-1 การออกแบบให้ทำงานในลักษณะของ High Availability

จากรูปที่ 4-1 นั้นเป็นการออกแบบให้ทำงานเป็น High Availability แบบที่ใช้ฮาร์ดแวร์น้อยที่สุดแล้ว

4.1.1 ส่วนประกอบของระบบ

ส่วนประกอบของระบบจะประกอบไปด้วย Real Server สองตัวทำหน้าที่ตรวจสอบกันและกันโดยการใช้ Heartbeat ในกรณีที่ Real Server ตัวที่ทำงานอยู่นั้นเกิดการล้มเหลว Real Server อีกตัวที่ทำงานในลักษณะของ Hot-Stan by ก็จะทำ IP address take-over และทำให้การให้บริการของระบบทำงานต่อไปได้อีก ตัว Real Server นั้นสามารถให้บริการได้หลายอย่างด้วย

4.1.2 High Availability แบบ Single Virtual Service



รูปที่ 4-2 การออกแบบในลักษณะของ Single Virtual Service

จากเอกสารนี้เราจะถือว่าการติดตั้งส่วนประกอบทางเครือข่ายนั้นเป็นไปอย่างถูกต้องแล้ว

4.1.2.1 Real Server

Heartbeat จะรันบน Real Server ทั้งสองตัว และจะคอยจัดการกับ Virtual address (Floating Address) ในการคอนฟิก Heartbeat นั้นให้เรานำไฟล์ต่างไปวางไว้ในไดเรคทอรีต่อไปนี้ /etc/ha.d/ha.cf /etc/ha.d/haresources และ /etc/ha.d/authkeys โดยที่ชื่อของ node ในไฟล์ /etc/ha.d/ha.cf และ /etc/ha.d/haresources จะต้องตรงกับค่าที่ส่งกลับออกมาจากคำสั่ง uname –n ส่วนไฟล์ /etc/ha.d/authkeys นั้นเราจะต้องทำการเปลี่ยนให้เป็นโหมด 600 ด้วยโดยการใช้คำสั่ง

```
chmod 600 /etc/ha.d/authkeys
```

และเพื่อเป็นการแน่ใจว่า Heartbeat นั้นจะถูกเรียกขึ้นมาใช้งานทันทีเมื่อระบบของเราบูต (on run-levels 2, 3, 4 และ 5) chkconfig ขึ้นมา สามารถทำได้โดยใช้คำสั่งดังต่อไปนี้

```
/sbin/chkconfig –level 2345 heartbeat on
```

คำสั่งนี้เป็นการอ้างอิงจาการทำงานบนลีนุกซ์ที่เป็น Red Hat 8.0 สำหรับค่ายอื่นๆ นั้นสามารถอ่านได้จาก www.ultramonkey.org และเพื่อเป็นการสั่งให้ Heartbeat ทำงานนั้นสามารถทำได้โดยใช้คำสั่ง

```
/etc/init.d/heartbeat start
```

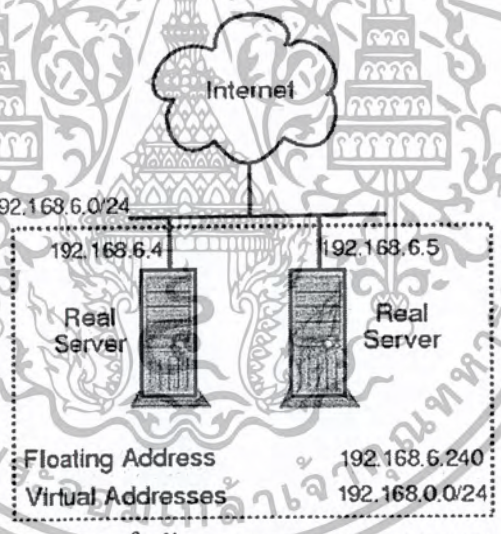
หลังจากที่เราทำการรันคำสั่งข้างต้นแล้ว อีกสักครู่ควรที่จะมีการสร้าง Virtual address บน Real Server ตัวแรกที่เราทำการพิมพ์คำสั่งเข้าไป ซึ่งเราสามารถตรวจสอบได้โดยคำสั่ง ifconfig ดังตัวอย่าง

```
/sbin/ifconfig
eth0:0    Link encap:Ethernet  HWaddr 00:90:27:6F:71:67
          inet addr:192.168.6.240  Bcast:192.168.6.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:18 Base address:0x1000
```

หมายเลขของ IP address ที่ได้ออกมานั้นอ้างอิงจากตัวอย่างในรูปที่ 4-2 และถ้าหากเกิดปัญหาอะไรขึ้นนั้นเราสามารถอ่านได้จาก log ไฟล์ซึ่งจะมีค่าดีฟอลต์ในการเก็บอยู่ที่ /var/log/messages

ตัวอย่างของไฟล์คอนฟิกทั้งหมดสามารถอ่านได้จากภาคผนวก ก.

4.1.3 High Availability แบบ Network of Virtual Services



รูปที่ 4-3 การออกแบบในลักษณะของ Network of Virtual Services

จากเอกสารนี้เราจะถือว่าการติดตั้งส่วนประกอบทางเครือข่ายนั้นเป็นไปอย่างถูกต้องแล้ว ค่าดีฟอลต์การ route สำหรับ Real Server นั้นให้สมมุติว่าเป็นเร้าเตอร์การออกแบบในลักษณะนี้อ้างอิงจากการออกแบบในลักษณะของ Single Virtual Service ดังนั้นการที่จะสร้าง Network of Virtual Services ได้นั้นจำเป็นที่จะต้องสร้าง Single Virtual Service ให้ได้เสียก่อน

4.1.3.1 การเตรียมเครือข่าย

สำหรับการออกแบบในลักษณะของ Network of Virtual Services นี้ เราจำเป็นที่ต้องทำการ rout เกตเว หรือ เร้าเตอร์ไปยัง Virtual service network โดยจากรูปที่ 4-3 จะเป็น 192.168.0.0/24 โดยผ่านทางหมายเลข IP ของ Virtual address คือ 192.168.7.240 สำหรับวิธีการทำนั้นสามารถทำได้ดังนี้

- ต้องทำการ Enable การส่งต่อแพคเกต IPV4 สามารถทำได้โดยการเพิ่มบรรทัดแหล่านี้เข้าไปในไฟล์ /etc/sysctl.conf

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
```

และเพื่อให้เกิดความเปลี่ยนแปลงจากคำสั่งที่เราได้เพิ่มเข้าไปให้รันคำสั่งนี้ /sbin/sysctl –p ซึ่งจะให้ผลลัพธ์ดังตัวอย่างต่อไปนี้

```
/sbin/sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
```

- การ Route 192.168.0.0/24 ไปที่ 192.168.6.240 นั้นสามารถทำได้โดยเพิ่มบรรทัดเหล่านี้เข้าไปในไฟล์ /etc/sysconfig/static-routes

```
any net 192.168.0.0 netmask 255.255.255.0 gw 192.168.6.240
```

และเพื่อให้เกิดความเปลี่ยนแปลงจากคำสั่งที่เราเพิ่มไปให้เรารันคำสั่งต่อไปนี้ /etc/init.d/network restart ซึ่งจะได้ผลลัพธ์ดังตัวอย่าง

```
/etc/init.d/network restart
Shutting down interface eth0:        [ OK ]
Setting network parameters:          [ OK ]
Bringing up interface lo:            [ OK ]
Bringing up interface eth0:          [ OK ]
```

โดยที่เราสามารถตรวจสอบการ Route ได้ดังตัวอย่าง

```
/sbin/ip route list 192.168.0.0/24
192.168.0.0/24 via 192.168.6.240 dev eth0
```

4.1.3.2 Real Server

ให้ทำตามขั้นตอนของการคอนฟิก Real Server ในรูปแบบของ Single Virtual Service ก่อนที่จะทำต่อในส่วนของ Network of Virtual Service

Real Server นั้นต้องทำการคอนฟิกเพื่อให้เห็นทราฟิกของ Virtual address ใน 192.168.0.0/24 ให้เป็นเหมือนกับเป็นโลคอล สามารถทำได้โดย การทำ IP alias บนอุปกรณ์ loopback โดยจากไฟล์ /etc/sysconfig/network-scripts/ifcfg-lo:0 ให้เราทำการคอนฟิกไฟล์ดังกล่าวดังตัวอย่างต่อไปนี้

```
DEVICE=lo:0
IPADDR=192.168.0.0
NETMASK=255.255.255.0
NETWORK=192.168.0.0
BROADCAST=192.168.0.255
ONBOOT=yes
NAME=loopback
```

และเพื่อเป็นการป้องกันการเกิดการเตือนโดยไม่จำเป็นให้เราแน่ใจว่าเราได้เซตอุปกรณ์เกตเวไปที่ eth0 เรียบร้อยแล้ว ให้เราแก้ไฟล์ /etc/sysconfig/network และเพิ่มบรรทัดดังตัวอย่างต่อไปนี้

```
GATEWAYDEV=eth0
```

และเพื่อทำการ Bring up IP alias นั้นให้เราใช้คำสั่ง ifup ดังตัวอย่าง

```
/sbin/ifup lo
```

เราสามารถตรวจสอบได้โดยการใช้คำสั่ง ifconfig ดังตัวอย่าง

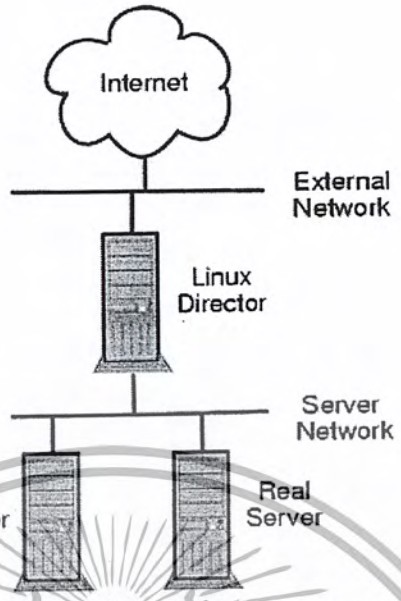
```
/sbin/ifconfig
lo:0      Link encap:Local Loopback
inet addr:192.168.0.0  Mask:255.255.255.0
UP LOOPBACK RUNNING  MTU:3924  Metric:1
```

ตัวอย่างของไฟล์คอนฟิกทั้งหมดสามารถอ่านได้จากภาคผนวก ก.

4.1.4 วิเคราะห์การออกแบบ High Availability

การทำงานในลักษณะนี้จะคล้ายกับการให้บริการของเซิร์ฟเวอร์ที่ให้บริการเพียงตัว จะแตกต่างก็ตรงที่ เมื่อตัวที่ให้บริการอยู่นั้นเกิดการล้มเหลว หรือต้องการนำไปซ่อมบำรุงก็สามารถทำได้เพราะมีเซิร์ฟเวอร์อีกตัวขึ้นมาทำหน้าที่แทนได้ แต่หากจะคิดกันถึงในเรื่องของความรวดเร็วของการบริการ และการ รองรับการเชื่อมต่อมากๆ ในกรณีที่มีผู้ใช้หลายๆ คนเข้ามาใช้การบริการในคราวเดียวกันแล้วนั้น การสร้างระบบในลักษณะนี้ จะไม่สามารถรองรับการงานลักษณะดังกล่าวได้เลย และอาจจะเกิดการล้มเหลวได้ง่ายอีกด้วย

4.2 การออกแบบให้ทำงานในแบบ Load Balancing



รูปที่ 4-4 การออกแบบให้ทำงานในแบบ Load Balancing

จากรูปที่ 4-4 เป็นการออกแบบระบบให้สามารถทำงานแบบ Load balancing ด้วยฮาร์ดแวร์ที่น้อยที่สุดแล้ว

4.2.1 ส่วนประกอบของระบบ

ระบบประกอบด้วยสองส่วนคือ Linux Director และ Real Server

- Linux Director

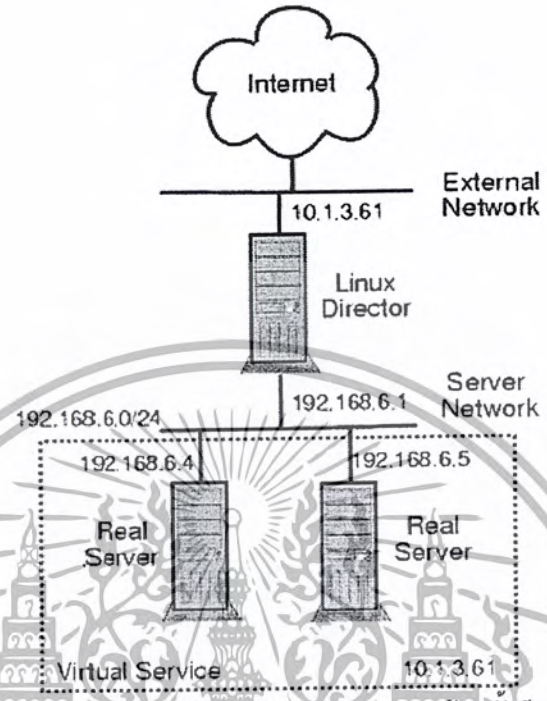
  จะทำหน้าที่เป็นเกตเวเร้าเตอร์ให้กับเครือข่ายของ Real Server โดยที่ตัว Linux Director นั้นจะต้องมีการ์ดแลนสองใบด้วยกัน (Network Interface Cards) จากรูป เมื่อเราทำการคอน ฟิก ให้ตัว Linux Director สามารถทำการส่งต่อแพคเกตได้แล้วนั้น เราก็ทำ NAT เพื่อเป็นการประหยัด IP address เมื่อตัว Linux Director ได้รับการเชื่อมต่อจากผู้ขอใช้บริการ ก็จะทำการตัดสินใจว่า Real Server ตัวใดในระบบที่จะต้องทำการส่งต่อแพคเกตไปให้ และจะต้องส่งแพคเกตไปที่ Real Server ตัวเดิมตลอดที่การเชื่อมต่อนั้นๆ ยังเกิดขึ้นอยู่ด้วย

- Real Server

  สามารถที่จะรันเซอร์วิสได้หลายเซอร์วิส

4.2.2 Load Balancing

จากรูปที่ 4-4 เมื่อเรานำมากำหนดหมาเลข IP ให้กับแต่ละเครื่องในระบบ ก็จะได้ดังรูปที่ 4-5



รูปที่ 4-5 การออกแบบให้ทำงานแบบ Load Balancing พร้อมทั้งมีการกำหนด IP

จากเอกสารนี้เราจะถือว่าการติดตั้งส่วนประกอบทางเครือข่ายนั้นเป็นไปอย่างถูกต้องแล้ว ส่วนของแพคเกตที่มีการส่งกลับนั้นจะต้องผ่านไปทาง Linux Director หมายความว่าค่าดีฟอลต์ของการเร้าควรจะเซ็ทไปที่การ์ดแลนใบที่ติดต่อกับเครือข่ายภายในของ Linux Director

4.2.2.1 Linux Director

- จะต้องสามารถเร้าการสื่อสารจากเครือข่ายด้านนอกเข้ามาที่เครือขายของเซิร์ฟเวอร์ด้านในได้ ต้องทำการ Enable การส่งต่อแพคเกต IPV4 สามารถทำได้โดยการเพิ่มบรรทัดแหล่านี้เข้าไปในไฟล์ /etc/sysctl.conf

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
```

และเพื่อให้เกิดความเปลี่ยนแปลงจากคำสั่งที่เราได้เพิ่มเข้าไปให้รันคำสั่งนี้ /sbin/sysctl –p ซึ่งจะให้ผลลัพธ์ดังตัวอย่างต่อไปนี้

```
/sbin/sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
```

การตรวจสอบ Real Server ไม่ว่าจะเป็นการเพิ่มรายการให้บริการของเซิร์ฟเวอร์ เพิ่มชื่อของเซิร์ฟเวอร์เข้าในส่วนของ Server Cluster จะถูกควบคุมโดย ldirectord ดังนั้นไฟล์ /etc/ha.d/ldirectord.cf จะต้องถูกติดตั้งลงไปด้วย และเพื่อเป็นการให้แน่ใจว่า ldirectord นั้นจะทำการสตาตขึ้นมาตอนที่ระบบของเราทำการบูท และ heartbeat จะไม่สตาตนั้นให้เราใช้คำสั่งดังต่อไปนี้

```
/sbin/chkconfig --level 2345 ldirectord on
/sbin/chkconfig --del heartbeat
```

เพื่อเป็นการให้แน่ใจว่า heartbeat ไม่ได้ทำงานและ ldirectord ทำงานอยู่ให้เรารันคำสั่งดังต่อไปนี้

```
/etc/init.d/heartbeat stop
/etc/init.d/ldirectord start
```

ตาราง Kernel ของ Linux Virtual Server จะแสดงได้โดยใช้คำสั่ง ipvsadm ดังตัวอย่าง ซึ่งจะบอกได้ว่าเครื่องใดใน Server Cluster ของเรากำลังให้บริการอะไรอยู่บ้าง

```
ipvsadm -L -n
IP Virtual Server version 1.0.4 (size=4096)
Prot LocalAddress:Port Scheduler Flags
 -> RemoteAddress:Port        Forward Weight ActiveConn InActConn
TCP  10.1.3.61:21 rr persistent 600
 -> 192.168.6.4:21            Masq    1      0          0
 -> 192.168.6.5:21            Masq    1      0          0
TCP  10.1.3.61:80 rr
 -> 192.168.6.4:80            Masq    1      0          0
 -> 192.168.6.5:80            Masq    1      0          0
```

```
TCP  10.1.3.61:443 rr
 -> 192.168.6.4:443              Masq    1      0          0
 -> 192.168.6.5:443              Masq    1      0          0
```

จากรูปที่เราออกแบบจะเห็นได้ว่าเป็นการทำ NAT ดังนั้นให้เราทำตามขั้นตอนดังต่อไปนี้

```
# Flush existing rules in the nat table
/etc/init.d/iptables stop
Resetting built-in chains to the default ACCEPT policy:     [ OK ]

# Masquerade for 192.168.6.0/24 bound for any host
/sbin/iptables -t nat -A POSTROUTING -j MASQUERADE -s 192.168.6.0/24

# Log all packets that attempt to be forwarded
# Useful for Debugging. Questionable for Production
#/sbin/iptables -t nat -A POSTROUTING -j LOG

# Save the rules
/etc/init.d/iptables save
Saving current rules to /etc/sysconfig/iptables:            [ OK ]

# Make sure rules are activated on reboot (at run levels 2, 3, 4 and 5)
/sbin/chkconfig --level 2345 iptables on

# Activate the rules
/etc/init.d/iptables start
Flushing all current rules and user defined chains:         [ OK ]
Clearing all current rules and user defined chains:         [ OK ]
Applying iptables firewall rules:                           [ OK ]
```

โดยที่เราสามารถตรวจสอบได้โดยใช้คำสั่งดังตัวอย่าง

```
/sbin/iptables -t nat -L -n
Chain POSTROUTING (policy ACCEPT)
```

target	port	opt	source	destination
MASQUERADE	all	--	192.168.6.0/24	0.0.0.0/0

4.2.2.2 Real Server

- การให้บริการบน Real Server นั้นจะต้องเป็นไปตามที่ตกลงไว้ในไฟล์ /etc/ha.d/ldirectord.cf เนื่องจากว่าเมื่อตัว Linux Director ทดสอบการใช้บริการเอง ก็จะสามารถทำได้ และเมื่อทำได้ก็จะประกาศว่ามีการให้บริการของการบริการดังกล่าวเข้าไปในตารางของ LVS
- จะเห็นได้ว่าเครือข่ายของเรานั้นมีการทำ NAT ดังนั้นการส่งกลับของแพคเกตจึงจะต้องผ่านไปให้กับตัว Linux Director ดังนั้นเราต้องเซทเกตเวของ Real Server ทุกๆ ตัวให้วิ่งไปที่ Linux Director ด้วย หากอ้างอิงจากรูปที่ 4-5 แล้วก็คือ IP หมายเลข 192.168.6.1 นั่นเอง โดยเราสามารถทำได้โดยการแก้ไขไฟล์ /etc/sysconfig/network-scripts/ifcfg-eth0 มีตัวอย่างดังนี้

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.6.4
NETMASK=255.255.255.0
GATEWAY=192.168.6.1
```

เมื่อแก้ไขเสร็จแล้วให้ทำการรีสตาสส่วนของเครือข่ายใหม่ด้วยโดยใช้คำสั่งดังตัวอย่าง

```
/etc/init.d/network restart
Shutting down interface eth0                [ OK ]
Setting network parameters                  [ OK ]
Bringing up interface lo                    [ OK ]
Bringing up interface eth0                  [ OK ]
```

เราสามารถตรวจสอบการเร้าได้โดย

```
/sbin/ip route show 0/0
default via 192.168.6.1 dev eth0
```

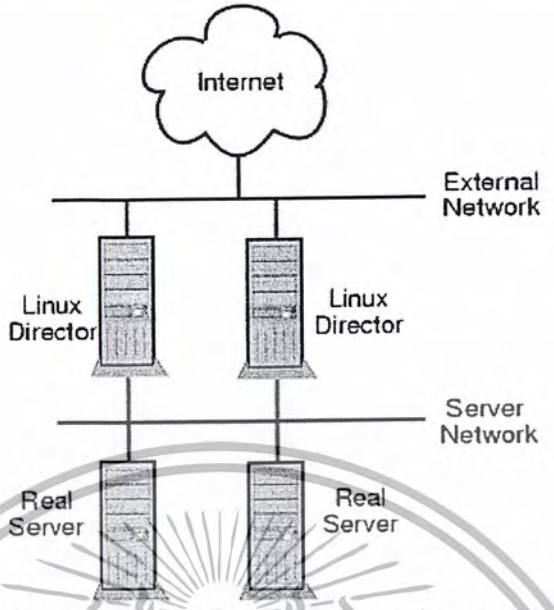
ตัวอย่างของไฟล์คอนฟิกทั้งหมดสามารถอ่านได้จากภาคผนวก ก.

4.2.3 วิเคราะห์การออกแบบ Load Balancing

การทำงานในลักษณะนี้เราจะเห็นได้ชัดเจนเลยว่ามีจุดล้มเหลวของระบบหนึ่งจุดก็คือตัว Linux Director เอง แม้นว่าระบบของเราซึ่งการให้บริการอยู่ที่ตัว Real Server นั้นจะสามารถทำงานได้เป็นปกติ แต่ถ้าหากว่าตัว Linux Director ซึ่งทำหน้าที่เป็นเกตเวให้กับระบบล้มเหลว หรือ ต้องการการซ่อมบำรุง แล้วละก็เป็นที่แน่นอนว่าจะทำให้การให้บริการทั้งหมดของระบบต้องหยุดลงเนื่องจากตัว Real Server จะไม่สามารถติดต่อกับภายนอกได้เลย

การทำงานในส่วนของ Real Server สองตัวนั้นมีลักษณะของการกระจายภาระการทำงาน ดังนั้นประสิทธิภาพที่ได้ จึงทำได้ดีกว่าแบบ High Availability อย่างแน่นอน และเมื่อการให้บริการของแต่ละ Real Server นั้นจะถูกตรวจสอบโดย Linux Director ดังนั้นหาก Real Server ตัวใดเกิดล้มเหลวขึ้น ตัว Linux Director ก็จะนำมันออกจาก Server Cluster ซึ่งก็หมายความว่า งานจะถูกส่งไปให้กับ Real Server ตัวที่เหลือทำ ถือว่าระบบมีความเป็น High Availability ในส่วนของ Real Server ด้วย

4.3 การออกแบบให้ทำงานในแบบ High Availability และ Load Balancing



รูปที่ 4-6 การออกแบบให้ทำงานในแบบ High Availability และ Load Balancing

จากรูปที่ 4-6 เป็นการออกแบบระบบให้สามารถทำงานแบบ High Availability และ Load Balancing ด้วยฮาร์ดแวร์ที่น้อยที่สุดแล้ว

4.3.1 ส่วนประกอบของระบบ

ระบบประกอบด้วยสองส่วนคือ Linux Director และ Real Server

- Linux Director

จากการออกแบบลักษณะที่แล้วเรามีตัว Linux Director เพียงตัวเดียวที่รับการเชื่อมต่อทั้งหมดจากผู้ขอใช้บริการซึ่งจะเห็นได้ว่า เกิดจุดล้มเหลวของระบบอยู่ แต่เมื่อเราออกแบบให้ทำงานในแบบดังรูปที่ 4-6 เราก็จะกำจัดจุดล้มเหลวของระบบออกไปได้ โดยตัว Linux Director นั้นจะมีการเปิดการทำงานของ Heartbeat ด้วยเพื่อเป็นการคอยตรวจตรากันและกัน โดยที่ Linux Director จะสร้าง Virtual IP address ขึ้นมาสอง IP ด้วยกันคือ IP ที่เอาไว้ติดต่อกับภายนอกและ IP ที่เอาไว้ทำติดต่อกับภายใน

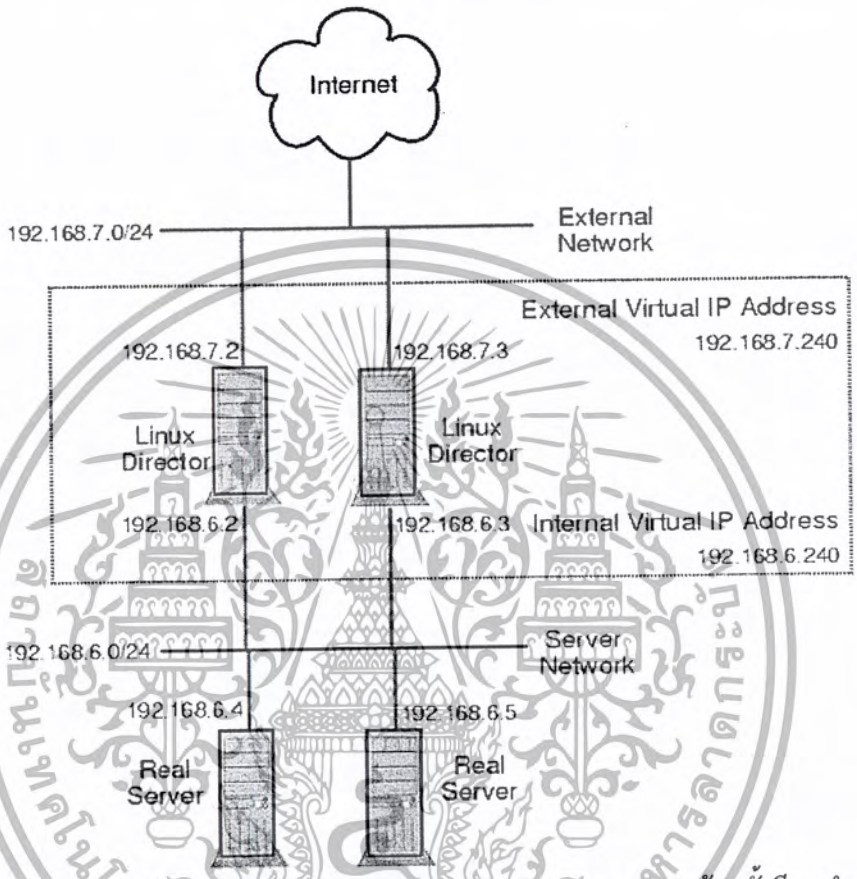
โดยที่ตัว Linux Director แต่ละตัวนั้นจะต้องมีการ์ดแลนสองใบด้วยกัน (Network Interface Cards) จากรูป เมื่อเราทำการคอนฟิกให้ตัว Linux Director สามารถทำการส่งต่อแพคเกตได้แล้วนั้น เราก็ทำ NAT เพื่อเป็นการประหยัด IP address เมื่อตัว Linux Director ได้รับการเชื่อมต่อจากผู้ขอใช้บริการ ก็จะทำการตัดสินใจว่า Real Server ตัวใดในระบบที่จะต้องทำการส่งต่อแพคเกตไปให้ และจะต้องส่งแพคเกตไปที่ Real Server ตัวเดิมตลอดที่การเชื่อมต่อนั้นๆยังเกิดขึ้นอยู่ด้วยตัว Linux Director มี Ldirectord ทำงานอยู่เพื่อคอยตรวจตรา Real Server ด้วย

- Real Server

  สามารถที่จะรันเซอร์วิสได้หลายเซอร์วิส

4.3.2 High Availability และ Load Balancing

จากรูปที่ 4-6 เมื่อเรานำมากำหนดหมาเลข IP ให้กับแต่ละเครื่องในระบบ ก็จะได้ดังรูปที่ 4-7



รูปที่ 4-7 การออกแบบให้ทำงานแบบ High Availability และ Load Balancing พร้อมทั้งมีการกำหนด IP

การออกแบบการเชื่อมต่อในลักษณะนี้จะสามารถทำงานอย่างสมบูรณ์แบบคือเป็นทั้ง High availability และเป็นแบบ Load balance ด้วย จำนวนเครื่องที่ต้องใช้อย่างน้อยคือ 4 เครื่อง

จากเอกสารนี้เราจะถือว่าการติดตั้งส่วนประกอบทางเครือข่ายนั้นเป็นไปอย่างถูกต้องแล้ว ส่วนของแพคเกตที่มีการส่งกลับนั้นจะต้องผ่านไปทาง Linux Director หมายความว่าค่าดีฟอลต์ของการเร้าจาก Real Server ควรจะเซ็ทไปที่ Internal Virtual IP Address

4.3.2.1 Linux Director

- จะต้องสามารถเร้าการสื่อสารจากเครือข่ายด้านนอกเข้ามาที่เครือขายของเซิร์ฟเวอร์ด้านในได้ ต้องทำการ Enable การส่งต่อแพคเกต IPV4 สามารถทำได้โดยการเพิ่มบรรทัดแหล่านี้เข้าไปในไฟล์ /etc/sysctl.conf

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
# Enables source route verification
net.ipv4.conf.default.rp_filter = 1
# Disables the magic-sysrq key
kernel.sysrq = 0
```

และเพื่อให้เกิดความเปลี่ยนแปลงจากคำสั่งที่เราได้เพิ่มเข้าไปให้รันคำสั่งนี้ /sbin/sysctl –p ซึ่งจะให้ผลลัพธ์ดังตัวอย่างต่อไปนี้

```
/sbin/sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 1
kernel.sysrq = 0
```

Heartbeat นั้นจะถูกติดตั้งไปที่ Linux Director ทั้งสองตัวเพื่อคอยทำการมอนิเตอร์ซึ่งกันและกัน และจะคอยจัดการกับการสร้าง Virtual IP สำหรับการคอนฟิก heartbeat นั้นให้เราติดตั้งไฟล์ลงไปที่ไดเรคทอรีดังต่อไปนี้ /etc/init.d/ha.cf, /etc/ha.d/haresources และ /etc/ha.d/authkeys โดยที่ชื่อของ node ในไฟล์ /etc/ha.d/ha.cf และ /etc/ha.d/haresources จะต้องตรงกับค่าที่ส่งกลับออกมาจากคำสั่ง uname –n ส่วนไฟล์ /etc/ha.d/authkeys นั้นเราจะต้องทำการเปลี่ยนให้เป็นโหมด 600 ด้วยโดยการใช้คำสั่ง

```
chmod 600 /etc/ha.d/authkeys
```

การตรวจสอบ Real Server ไม่ว่าจะเป็นการเพิ่มรายการให้บริการของเซิร์ฟเวอร์ เพิ่มชื่อของเซิร์ฟเวอร์เข้าในส่วนของ Server Cluster จะถูกควบคุมโดย ldirectord ดังนั้นไฟล์ /etc/ha.d/ldirectord.cf จะต้องถูกติดตั้งลงไปด้วยและเพื่อเป็นการแน่ใจว่า Heartbeat นั้นจะถูกเรียกขึ้นมาใช้งานทันทีเมื่อระบบของเราบูต (on run-levels 2, 3, 4 และ 5) chkconfig ขึ้นมา

และ ldirectord จะต้องไม่ถูกเรียกขึ้นมาจนกว่า Heartbeat จะทำงาน สามารถทำได้โดยใช้คำสั่งดังต่อไปนี้

```
/sbin/chkconfig –level 2345 heartbeat on
/sbin/chkconfig –del ldirectord
```

เพื่อเป็นการให้แน่ใจว่า ldirectord ไม่ได้ทำงานและ heartbeat ทำงานอยู่ให้เรารันคำสั่งดังต่อไปนี้

```
/etc/init.d/heartbeat stop
/etc/init.d/ldirectord start
```

หลังจากที่เราทำการรันคำสั่งข้างต้นแล้ว อีกสักครู่ควรที่จะมีการสร้าง Virtual address บน Linux Director ตัวแรกที่เราทำการพิมพ์คำสั่งเข้าไป ซึ่งเราสามารถตรวจสอบได้โดยคำสั่ง ifconfig ดังตัวอย่าง

```
/sbin/ifconfig
eth0:0    Link encap:Ethernet  HWaddr 00:D0:B7:BE:6B:CF
          inet addr:192.168.6.240  Bcast:192.168.6.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:17 Base address:0xef00

eth1:0    Link encap:Ethernet  HWaddr 00:90:27:74:84:ED
          inet addr:192.168.7.240  Bcast:192.168.7.355 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:18 Base address:0xee80
```

หมายเลขของ IP address ที่ได้ออกมานั้นอ้างอิงจากตัวอย่างในรูปที่ 4-7 และถ้าหากเกิดปัญหาอะไรขึ้นนั้นเราสามารถอ่านได้จาก log ไฟล์ซึ่งจะมีค่าดีฟอลต์ในการเก็บอยู่ที่ /var/log/messages

Heartbeat ที่เราทำการสตาตขึ้นมานั้นควรที่จะไปเรียก ldirectord ให้ขึ้นมาทำงานโดยเราสามารถตรวจสอบได้โดย

```
/usr/sbin/ldirectord ldirectord.cf status
ldirectord for ldirectord.cf is running with pid: 30314
```

เราสามารถตรวจสอบตาราง LVS Kernel ได้โดยการใช้คำสั่ง ipvsadm ดังตัวอย่าง ซึ่งจะบอกได้ว่าเครื่องใดใน Server Cluster ของเรากำลังให้บริการอะไรอยู่บ้าง

```
/sbin/ipvsadm -L -n
IP Virtual Server version 0.9.16 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.7.240:443 rr
  -> 192.168.6.4:443              Masq    1      0          0
  -> 192.168.6.5:443              Masq    1      0          0
TCP  192.168.7.240:80 rr
  -> 192.168.6.4:80               Masq    1      0          0
  -> 192.168.6.5:80               Masq    1      0          0
TCP  192.168.7.240:21 rr
  -> 192.168.6.4:21               Masq    1      0          0
  -> 192.168.6.5:21               Masq    1      0          0
```

Linux Director ตัวที่ไม่ได้ทำการรัน Heartbeat ขณะนั้นควรที่จะ stand-by และจะต้องทำงานทันทีที่เราสั่งให้หยุด Heartbeat ในเครื่องแรกที่เราทำการรัน สามารถทดสอบได้โดยลองหยุดการทำงานของ Heartbeat ในเครื่องที่ทำการรันอยู่ด้วยคำสั่งต่อไปนี้

```
/etc/init.d/heartbeat stop
```

จากรูปที่เราออกแบบจะเห็นได้ว่าเป็นการทำ NAT ดังนั้นให้เราทำตามขั้นตอนดังต่อไปนี้

```
# Flush existing rules in the nat table
/etc/init.d/iptables stop
Resetting built-in chains to the default ACCEPT policy:      [ OK ]

# Masquerade for 192.168.6.0/24 bound for any host
/sbin/iptables -t nat -A POSTROUTING -j MASQUERADE -s 192.168.6.0/24

# Log all packets that attempt to be forwarded
# Useful for Debugging. Questionable for Production
#/sbin/iptables -t nat -A POSTROUTING -j LOG

# Save the rules
/etc/init.d/iptables save
Saving current rules to /etc/sysconfig/iptables:      [ OK ]

# Make sure rules are activated on reboot (at run levels 2, 3, 4 and 5)
/sbin/chkconfig --level 2345 iptables on

# Activate the rules
/etc/init.d/iptables start
Flushing all current rules and user defined chains:      [ OK ]
Clearing all current rules and user defined chains:      [ OK ]
Applying iptables firewall rules:                        [ OK ]
```

โดยที่เราสามารถตรวจสอบได้โดยใช้คำสั่งดังตัวอย่าง

```
/sbin/iptables -t nat -L -n
Chain POSTROUTING (policy ACCEPT)
target          port    opt     source          destination
MASQUERADE      all     --      192.168.6.0/24  0.0.0.0/0
```

ตัวอย่างของไฟล์คอนฟิกทั้งหมดสามารถอ่านได้จากภาคผนวก ก.

4.3.2.2 Real Server

- การให้บริการบน Real Server นั้นจะต้องเป็นไปตามที่ตกลงไว้ในไฟล์ /etc/ha.d/ldirectord.cf เนื่องจากว่าเมื่อตัว Linux Director ทดสอบการใช้บริการเอง ก็จะสามารถทำได้ และเมื่อทำได้ก็จะประกาศว่ามีการให้บริการของการบริการดังกล่าวเข้าไปในตารางของ LVS
- จะเห็นได้ว่าเครือข่ายของเรานั้นมีการทำ NAT ดังนั้นการส่งกลับของแพคเกตจึงจะต้องผ่านไปให้กับตัว Linux Director ดังนั้นเราต้องเซทเกตเวของ Real Server ทุกๆ ตัวให้วิ่งไปที่ Virtual address ที่ Linux Director สร้างขึ้น หากอ้างอิงจากรูปที่ 4-6 แล้วก็คือ IP หมายเลข 192.168.6.240 นั่นเอง โดยเราสามารถทำได้โดยการแก้ไขไฟล์ /etc/sysconfig/network-scripts/ifcfg-eth0 มีตัวอย่างดังนี้

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.6.4
NETMASK=255.255.255.0
GATEWAY=192.168.6.240
```

เมื่อแก้ไขเสร็จแล้วให้ทำการรีสตาสส่วนของเครือข่ายใหม่ด้วยโดยใช้คำสั่งดังตัวอย่าง

```
/etc/init.d/network restart
Shutting down interface eth0                [  OK  ]
Setting network parameters                  [  OK  ]
Bringing up interface lo                    [  OK  ]
Bringing up interface eth0                  [  OK  ]
```

เราสามารถตรวจสอบการเร้าได้โดย

```
/sbin/ip route show 0/0
default via 192.168.6.240 dev eth0
```

4.3.3 วิเคราะห์การออกแบบ High Availability และ Load Balancing

เมื่อเราออกแบบระบบให้ทำงานแบบนี้แล้ว ตัว Linux Director ของเราก็สามารถทำงานได้ครบทุกอย่างที่เราต้องการ คือมีความสามารถทั้งทางด้านการกระจายภาระการทำงานให้กับ Real Server แต่ละตัวในระบบ ทั้งยังมีความสามารถในเชิง High Availability อีกด้วย คือเมื่อ Linux Director ตัวใดตัวหนึ่งเกิดล้มเหลว หรือต้องการปิดเพื่อทำการซ่อมบำรุง อีกตัวที่เหลือก็จะสามารถทำหน้าที่ต่อได้

การทำงานในส่วนของ Real Server สองตัวนั้นมีลักษณะของการกระจายภาระการทำงาน ดังนั้นประสิทธิภาพที่ได้ จึงทำได้ดีกว่าแบบ High Availability อย่างแน่นอน และเมื่อการให้บริการของแต่ละ Real Server นั้นจะถูกตรวจสอบโดย Linux Director ดังนั้นหาก Real Server ตัวใดเกิดล้มเหลวขึ้น ตัว Linux Director ก็จะนำมันออกจาก Server Cluster ซึ่งก็หมายความว่า งานจะถูกส่งไปให้กับ Real Server ตัวที่เหลือทำ ถือว่าระบบมีความเป็น High Availability ในส่วนของ Real Server ด้วย

บทที่ 5
บทวิจารณ์และสรุป

5.1 สรุปผลการดำเนินงาน

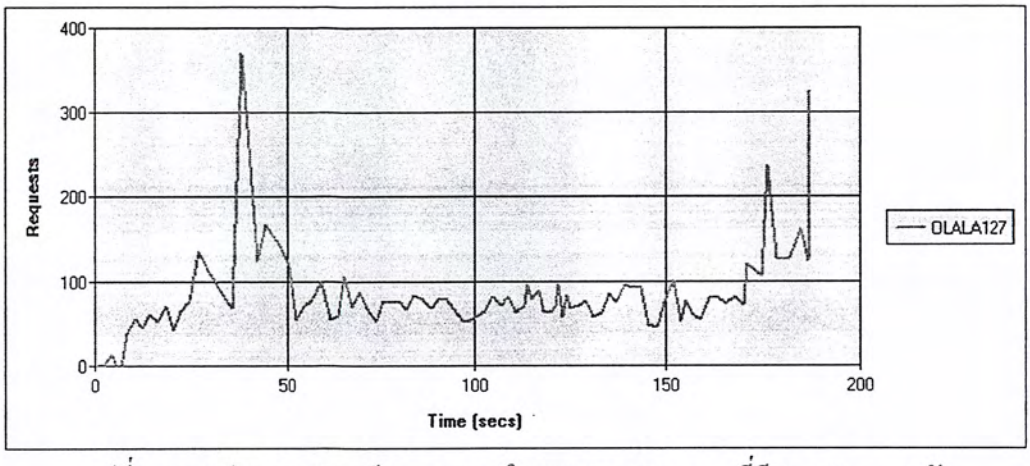
เมื่อเราพิจารณาถึงด้านความมีเสถียรภาพของระบบแล้ว เราจะพิจารณาจากสองการออกแบบคือ High Availability และ High Availability and Load Balancing จากการทดสอบเปิดให้บริการเป็นเวลา 1 อาทิตย์พบว่าไม่มีระบบใดเกิดความล้มเหลวเกิดขึ้น

เมื่อเราพิจารณาถึงด้านประสิทธิภาพของการให้บริการพบว่าจากการติดตั้งระบบและทดสอบการทำงานในทุกๆ แบบที่เราได้กล่าวมาทั้งหมดแล้วนั้น เมื่อนำมาเปรียบเทียบกับ Real Server เครื่องเดี่ยวๆ ที่ให้บริการ Web Server และ ระบบของเราก็ให้บริการเช่นเดียวกันสามารถสรุปผลได้ดังตารางต่อไปนี้

วิธีการออกแบบ	RPS
Real Server เครื่องเดียว	102.53
High Availability (2 real servers)	98.3
Load Balance (2 real servers)	128.83

ตารางที่ 5-1 การเปรียบเทียบประสิทธิภาพการทำงานในแต่ละวิธีการออกแบบ

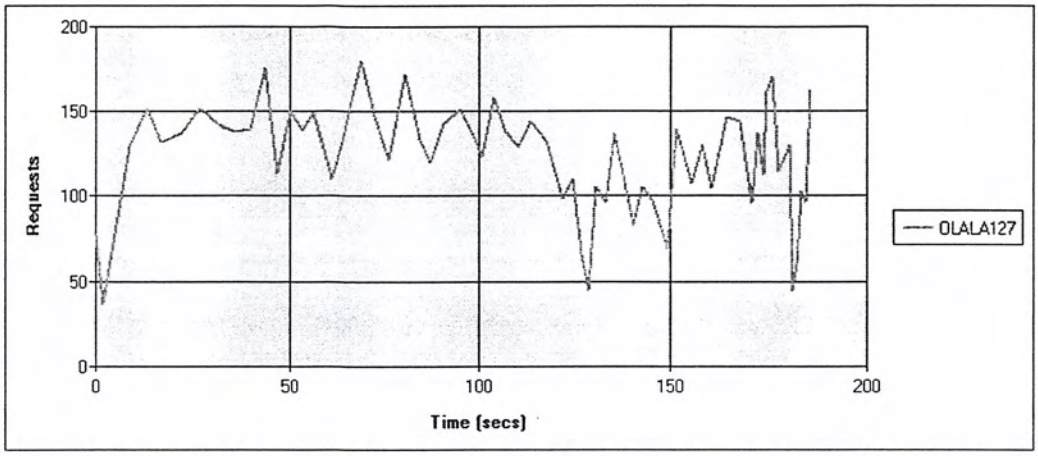
จากตารางที่ 5-1 เราจะเห็นว่า Real Server เครื่องเดียวนั้น สามารถทำงานได้สูสีกับระบบของเราที่เป็นแบบ High Availability เนื่องจากว่ามีโครงสร้างการทำงานเหมือนกัน เพราะว่าไม่ต้องทำการส่งผ่านแพคเกตกลับไปที่ Linux Director และจากการทดลองยังพบอีกว่า ระบบที่ออกแบบให้เป็นแบบ Load Balance หากเหลือ Real Server ที่ให้บริการอยู่เพียงตัวเดียวนั้น ยังทำงานได้ช้ากว่า Real Server ตัวเดี่ยวๆ เสียอีก นั่นเป็นเพราะว่า การที่ต้องมีการเขียนแพคเกตเพื่อส่งใหม่นั่นเอง แต่ถ้าหากเราให้ระบบของเราทำงานในแบบ Load Balance และเพิ่ม Real Server ที่ให้บริการเป็น 2 ตัว เราจะได้ค่าของ RPS (request per second) ที่สูงขึ้นกว่าเดิม



รูปที่ 5-1 กราฟของ RPS กรณีการออกแบบในแบบ Load Balance ที่มี Real Server 1 ตัว

รายละเอียดอื่นๆ ของการออกแบบในแบบ Load Balance ที่มี Real Server 1 ตัว
จากการทดสอบจำลองผู้ใช้งาน 200 คน ภายในเวลา 3 นาที

Summary	
Total number of requests:	15,437
Total number of connections:	15,325
* * **Average requests per second:**	**85.76** * *
Average time to first byte (msecs):	1,381.94
Average time to last byte (msecs):	1,952.03
Average time to last byte per iteration (msecs):	1,919.57
Number of unique requests made in test:	1
Number of unique response codes:	1
Errors Counts	
HTTP:	15,437
DNS:	0
Socket:	73

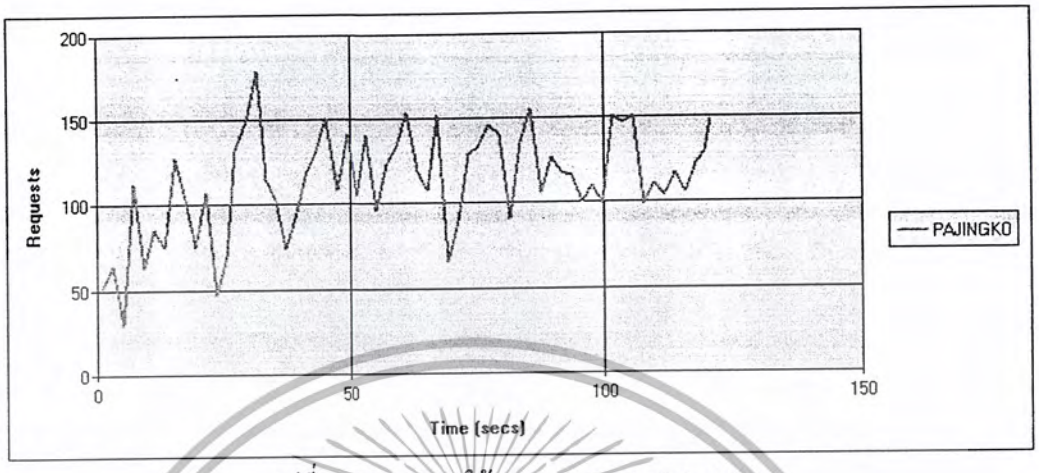


รูปที่ 5-2 กราฟของ RPS กรณีการออกแบบในแบบ Load Balance ที่มี Real Server 2 ตัว

รายละเอียดอื่นๆ ของการออกแบบในแบบ Load Balance ที่มี Real Server 2 ตัว
จากการทดสอบจำลองผู้ใช้งาน 200 คน ภายในเวลา 3 นาที

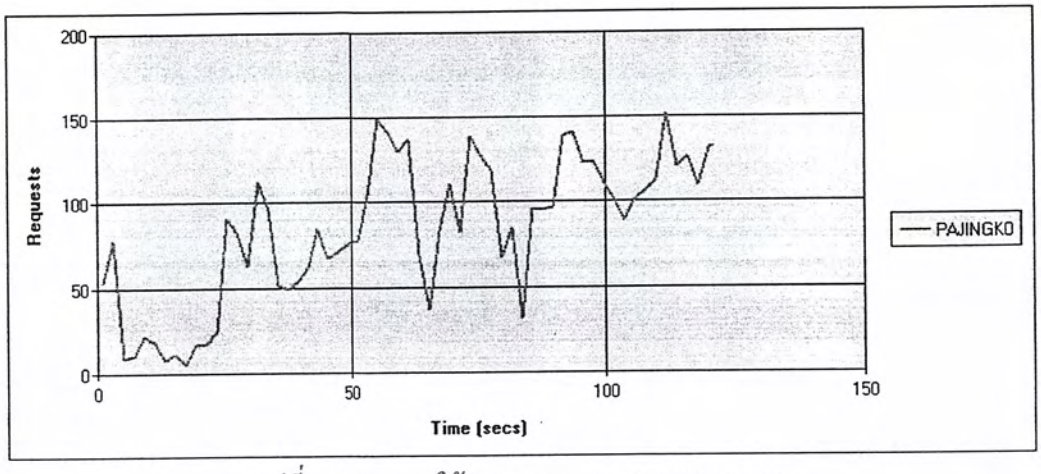
Summary	
Total number of requests:	23,190
Total number of connections:	23,167
****Average requests per second:**	**128.83****
Average time to first byte (msecs):	518.36
Average time to last byte (msecs):	647.09
Average time to last byte per iteration (msecs):	638.63
Number of unique requests made in test:	1
Number of unique response codes:	1
Errors Counts	
HTTP:	23,190
DNS:	0
Socket:	30

การพิจารณาประสิทธิภาพนั้นสามารถวัดได้จากลักษณะของการเชื่อมต่อแล้ว ยังสามารถวัดได้จากวิธีการเลือกใช้ Scheduling Algorithm ในการเลือก Real Server ขึ้นมาทำงาน การเลือกใช้ Scheduling Algorithm ในแต่ละแบบสามารถดูได้จากผลของการทดสอบ



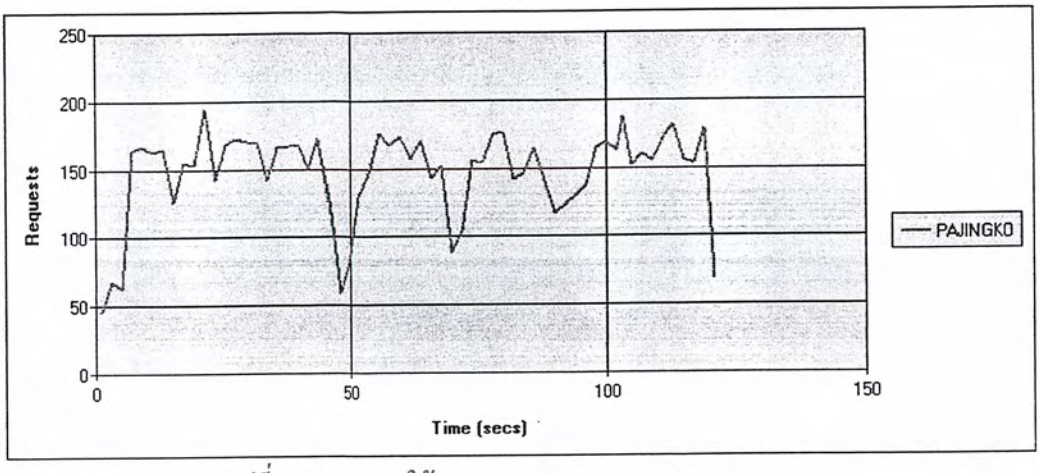
รูปที่ 5-3 ผลการจากใช้ Round – Robin Scheduling

Properties	
Test type:	Dynamic
Simultaneous browser connections:	1,000
Warm up time (secs):	0
Test duration:	00:00:02:00
Test iterations:	36,369
Detailed test results generated:	Yes
Summary	
Total number of requests:	13,697
Total number of connections:	14,221
Average requests per second:	114.14
Average time to first byte (msecs):	2,040.42
Average time to last byte (msecs):	2,148.48
Average time to last byte per iteration (msecs): 809.14	
Number of unique requests made in test:	1
Number of unique response codes:	2
Errors Counts	
HTTP:	82
DNS:	0
Socket:	22,744



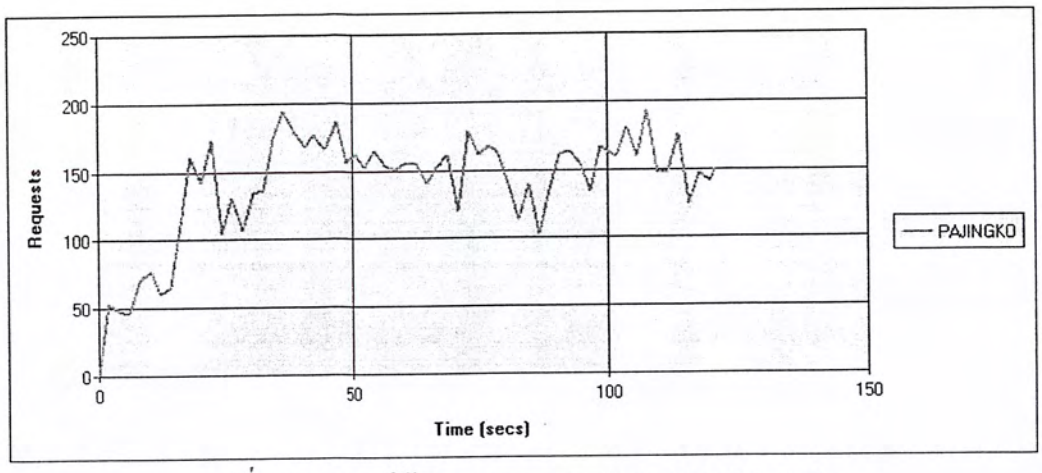
รูปที่ 5-4 ผลจากการใช้ Least – Connection Scheduling

Properties	
Test type:	Dynamic
Simultaneous browser connections:	1,000
Warm up time (secs):	0
Test duration:	00:00:02:00
Test iterations:	31,615
Detailed test results generated:	Yes
Summary	
Total number of requests:	10,182
Total number of connections:	10,657
Average requests per second:	84.85
Average time to first byte (msecs):	1,955.03
Average time to last byte (msecs):	2,039.68
Average time to last byte per iteration (msecs):	656.90
Number of unique requests made in test:	1
Number of unique response codes:	2
Errors Counts	
HTTP:	23
DNS:	0
Socket:	21,484



รูปที่ 5-5 ผลจากการใช้ Weighted Round – Robin Scheduling

Properties	
Test type:	Dynamic
Simultaneous browser connections:	1,000
Warm up time (secs):	0
Test duration:	00:00:02:00
Test iterations:	34,728
Detailed test results generated:	Yes
Summary	
Total number of requests:	17,812
Total number of connections:	19,207
Average requests per second:	148.43
Average time to first byte (msecs):	2,116.01
Average time to last byte (msecs):	2,275.37
Average time to last byte per iteration (msecs):	1,167.04
Number of unique requests made in test:	1
Number of unique response codes:	2
Errors Counts	
HTTP:	18
DNS:	0
Socket:	17,022



รูปที่ 5-6 ผลจากการใช้ Weighted Least – Connection Scheduling

Properties	
Test type:	Dynamic
Simultaneous browser connections:	1,000
Warm up time (secs):	0
Test duration:	00:00:02:00
Test iterations:	33,966
Detailed test results generated:	Yes
Summary	
Total number of requests:	17,140
Total number of connections:	17,894
Average requests per second:	142.83
Average time to first byte (msecs):	2,361.63
Average time to last byte (msecs):	2,492.49
Average time to last byte per iteration (msecs):	1,257.77
Number of unique requests made in test:	1
Number of unique response codes:	2
Errors Counts	
HTTP:	140
DNS:	0
Socket:	16,878

จากผลการทดสอบจะเห็นได้ว่าวิธีการที่ใช้ในการเลือก Real Server ขึ้นมาทำงานนั้น สามารถสรุปได้ดังตารางที่ 5-2

ผลที่ต้องการ	วิธีการ
Average requests per second มากที่สุด	Weighted Round – Robin Scheduling
HTTP Error น้อยที่สุด	Weighted Round – Robin Scheduling
Socket Error น้อยที่สุด	Weighted Least – Connection Scheduling

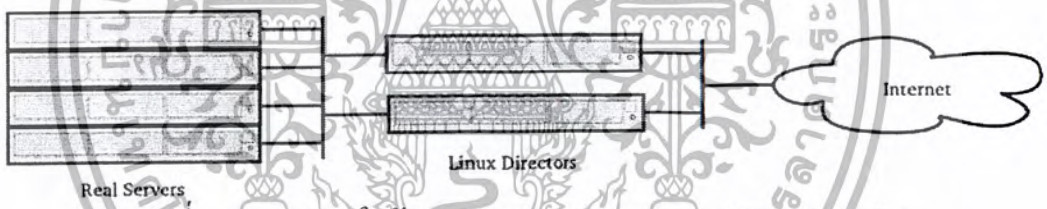
ตารางที่ 5-2 สรุปผลของวิธีการเลือก Real Server ขึ้นมาทำงานแบ่งตามผลที่ต้องการ

การวัดประสิทธิภาพของระบบทั้งหมดนั้นใช้เครื่องมือคือ Microsoft Application Center Test เฉพาะการวัดประสิทธิภาพของ Scheduling Algorithm เท่านั้นที่ใช้ 1 Linux Director และ Real Server 3 ตัว การวัดประสิทธิภาพอื่นๆ ที่ได้ทำการทดลองได้ใช้อุปกรณ์ที่น้อยที่สุดของการออกแบบนั้นๆ

5.2 แนวทางการพัฒนางานต่อ

จากระบบที่เราได้ทดลองสร้างขึ้นมานั้นโดยใช้หลักการของ Linux Virtual Server และ High Availability นั้นสามารถให้บริการได้อย่างมีประสิทธิภาพระดับหนึ่งแล้ว แต่สิ่งที่ระบบนี้ยังขาดอยู่คือ

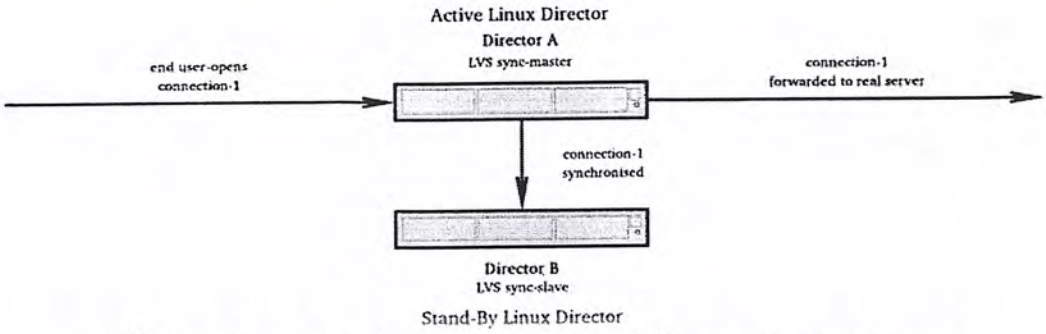
5.2.1. Connection Synchronization



รูปที่ 5-7 การออกแบบในลักษณะของ High Availability and Load Balancing

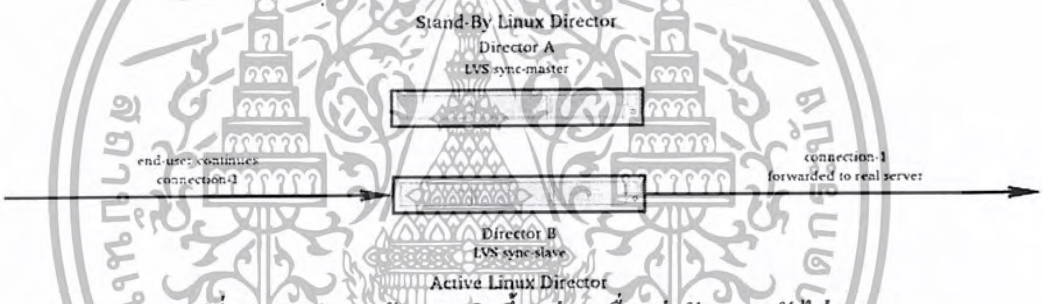
จากการที่ผู้ขอใช้บริการเข้ามาติดต่อกับระบบและในขณะที่ใช้บริการระบบเกิดการล้มเหลว เกิดขึ้นการเชื่อมต่อของผู้ใช้บริการคนนั้นจะค้าง เนื่องจากแพคเกตที่กำลังส่งอยู่ระหว่างการให้บริการนั้นจะอยู่กับ Linux Director ตัวที่เกิดการล้มเหลวขึ้น

โดยที่เราสามารถแก้ไขปัญหาดังกล่าวได้โดยแนวคิดของ Connection Synchronization ซึ่งมีแนวคิดดังนี้คือ ให้มีการเลือก Master node ขึ้นระหว่า Linux Director ทั้งสองตัว



รูปที่ 5-8การส่งแพคเกตระหว่า Active Linux Director ให้กับ Stand-By Linux Director

เมื่อผู้ขอใช้บริการเริ่มสร้างการติดต่อกับระบบ และเมื่อเครื่อง Linux Director ตัวที่ถือครอง Virtual IP อยู่นั้นได้รับแพคเกตจากผู้ขอใช้บริการ ก็จะทำการส่งให้กับเครื่องที่เป็น Linux Director ตัวที่ Stand-By อยู่ด้วย และส่งต่อแพคเกตไปให้กับ Real Server



รูปที่ 5-9 การเกิดการล้มเหลวเกิดขึ้นแต่การเชื่อมต่อกับระบบยังไม่ขาด

เมื่อเกิดการล้มเหลวเกิดขึ้นนั้น Slave Linux Director ก็จะทำหน้าที่แทนตัวที่เป็น Master โดยการถือครอง Virtual IP และจะเปลี่ยนตัวเองไปอยู่ในลักษณะ Active แต่การเชื่อมต่อกับระบบจะยังไม่ขาดหายไป เนื่องมาจาก การที่ตัว Slave node นั้นได้รับแพคเกตจาก Master node ทุกครั้งที่มีแพคเกตเข้ามาในระบบ

5.2.1. Distributed File System

หาเราลองคิดถึงการใช้บริการ FTP แล้วหละก็ เมื่อเครื่อง Linux Director ตัดสินใจว่าเราจะนำไฟล์ไปเก็บไว้ที่ Real Server ตัวใดในครั้งแรกของการเชื่อมต่อแล้ว แต่ในการเชื่อมต่อครั้งที่สอง Linux Director ไม่ได้ตัดสินใจให้เราติดต่อกับ Real Server ตัวเดิมแล้วไฟล์ที่เรานำไปเก็บไว้ใน Real Server เครื่องแรกเราก็จะหาไม่พบ และถ้าหากเราใช้ NFS มาแก้ปัญหาลักษณะนี้คือให้ Real Server ที่อยู่ในระบบของเราทำการ mount ไดเรคทอรี NFS เซิร์ฟเวอร์แล้วก็จะแก้ปัญหาลักษณะนี้ได้ แต่ ระบบของเราจะไม่สามารถทำงานในแบบ High Availability ได้ทันที่เนื่องมาจากจุดอ่อนของระบบเราได้ไปอยู่ NFS เซิร์ฟเวอร์แล้ว แม้นว่าการให้บริการยังสามารถทำได้แต่ไม่มีไฟล์ให้ผู้ใช้บริการได้ใช้ ก็ไม่ถือว่าเป็นการ

ให้บริการที่สมบรูณ์ ดังนั้นแนวทางในการแก้ปัญหาลักษณะนี้คือ ศึกษาถึงแนวคิดและวิธีใช้งานของ Distributed File System เพื่อนำมาแก้ปัญหาในลักษณะนี้ ตัวอย่างเช่น GFS , Coda หรือ Intermezzo ซึ่งระบบไฟล์เหล่านี้จะคำนึงถึงเรื่อง Availability กับ Scalability ของการเข้าถึงไฟล์ของระบบ ซึ่งเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์จะทำการเข้าถึงระบบไฟล์เหมือนกับการเข้าถึงไฟล์ที่เครื่องของตนเอง ดังที่ได้กล่าวไว้แล้วในบทที่ 2

ภาคผนวก

ภาคผนวก ก.

ตัวอย่างไฟล์คอนฟิกของการออกแบบในลักษณะ High Availability แบบ Single Virtual Service และ Network of Virtual Service

/etc/ha.d/ha.cf

```
#
# /etc/ha.d/ha.cf
#
# ha.cf file to configure two nodes connected by ethernet (eth0) and
# a null modem (/dev/ttyS0).
#
# Based on sample ha.cf shipped with heartbeat
#
# Prepared: March 2003
#


#
#       There are lots of options in this file.  All you have to have is a set
#       of nodes listed {"node ...}
#       and one of {serial, bcast, mcast, or ucast}
#
#
#    Note on logging:
#    If any of debugfile, logfile and logfacility are defined then they
#    will be used. If debugfile and/or logfile are not defined and
#    logfacility is defined then the respective logging and debug
#    messages will be loged to syslog. If logfacility is not defined
#    then debugfile and logfile will be used to log messges. If
#    logfacility is not defined and debugfile and/or logfile are not
#    defined then defaults will be used for debugfile and logfile as
#    required and messages will be sent there.
```

```
#
#       File to write debug messages to
#debugfile /var/log/ha-debug
#
#
#       File to write other messages to
#
#logfile    /var/log/ha-log
#
#
#       Facility to use for syslog()/logger
#
logfacility  local0
#
#
#       A note on specifying "how long" times below...
#
#       The default time unit is seconds
#               10 means ten seconds
#
#       You can also specify them in milliseconds
#               1500ms means 1.5 seconds
#
#
#       keepalive: how long between heartbeats?
#
keepalive 2
#
#       deadtime: how long-to-declare-host-dead?
#
deadtime 10
#
#       warntime: how long before issuing "late heartbeat" warning?
#       See the FAQ for how to use warntime to tune deadtime.
```

```
#
warntime 10
#
#
#       Very first dead time (initdead)
#
#       On some machines/OSes, etc. the network takes a while to come up
#       and start working right after you've been rebooted.  As a result
#       we have a separate dead time for when things first come up.
#       It should be at least twice the normal dead time.
#
initdead 10
#
#
#       nice_failback:  determines whether a resource will
#       fail back to its "primary" node, or remain on whatever
#       node is serving it until that node fails.
#
#       The default is "off", which means that it WILL fail
#       back to the node which is declared as primary in haresources
#
#       "on" means that resources only move to new nodes when
#       the nodes they are served on die.  This is deemed as a
#       "nice" behavior (unless you want to do active-active).
#
nice_failback on
#
#       hopfudge maximum hop count minus number of nodes in config
#hopfudge 1
#
#       serial  serialportname ...
serial  /dev/ttyS0  # Linux
#serial  /dev/cuaa0 # FreeBSD
#serial  /dev/cua/a  # Solaris
```

```
#
#
#       Baud rate for serial ports...
#
#baud   19200
#
#       What UDP port to use for communication?
#
#udpport 694
#
#       What interfaces to heartbeat over?
#
#bcast  eth0            # Linux
#bcast  le0             # Solaris
#
#       Set up a multicast heartbeat medium
#       mcast [dev] [mcast group] [port] [ttl] [loop]
#
#               [dev]           device to send/rcv heartbeats on
#               [mcast group]   multicast group to join (class D multicast address
#                               224.0.0.0 - 239.255.255.255)
#               [port]          udp port to sendto/rcvfrom (no real reason to differ
#                               from the port used for broadcast heartbeats)
#               [ttl]           the ttl value for outbound heartbeats.  this effects
#                               how far the multicast packet will propagate.  (0-255)
#               [loop]          toggles loopback for outbound multicast heartbeats.
#                               if enabled, an outbound packet will be looped back and
#                               received by the interface it was sent on. (0 or 1)
#
#
mcast eth0 225.0.0.7 694 1 1
#
#       Watchdog is the watchdog timer.  If our own heart doesn't beat for
#       a minute, then our machine will reboot.
```

```
#
#watchdog /dev/watchdog
#
#    "Legacy" STONITH support
#    Using this directive assumes that there is one stonith
#    device in the cluster.  Parameters to this device are
#    read from a configuration file. The format of this line is:
#
#     stonith <stonith_type> <configfile>
#
#    NOTE: it is up to you to maintain this file on each node in the
#    cluster!
#
#stonith baytech /etc/ha.d/conf/stonith.baytech
#
#    STONITH support
#    You can configure multiple stonith devices using this directive.
#    The format of the line is:
#     stonith_host <hostfrom> <stonith_type> <params...>
#     <hostfrom> is the machine the stonith device is attached
#        to or * to mean it is accessible from any host.
#     <stonith_type> is the type of stonith device (a list of
#        supported drives is in /usr/lib/stonith.)
#     <params...> are driver specific parameters.  To see the
#        format for a particular device, run:
#      stonith -l -t <stonith_type>
#
#
#           Note that if you put your stonith device access information in
#           here, and you make this file publically readable, you're asking
#           for a denial of service attack ;-)
#
#
#stonith_host *    baytech 10.0.0.3 mylogin mysecretpassword
```

```
#stonith_host ken3  rps10 /dev/ttyS1 kathy 0
#stonith_host kathy rps10 /dev/ttyS1 ken3 0
#
#       Tell what machines are in the cluster
#       node    nodename ...    -- must match uname -n
node    fred
node    mary
#
#       Less common options...
#
#       Treats 10.10.10.254 as a psuedo-cluster-member
#
#ping 10.10.10.254
#
#       Started and stopped with heartbeat.  Restarted unless it exits
#                                       with rc=100
#
#respawn userid /path/name/to/run
```

/etc/authkeys

```
#
# /etc/ha.d/authkeys
#
# Sample authkeys file. You should change the key and set the mode
# to 600
#
# Based on sample ha.cf shipped with heartbeat
#
# Prepared: July 2002
#
#
#       Authentication file.  Must be mode 600
#       Must have exactly one auth directive at the front.
```

```
#       auth        send authentication using this method-id
#       Then, list the method and key that go with that method-id
#       Available methods: crc sha1, md5.  Crc doesn't need/want a key.
#       You normally only have one authentication method-id listed in this file
#       Put more than one to make a smooth transition when changing auth
#       methods and/or keys.
#       sha1 is believed to be the "best", md5 next best.
#       crc adds no security, except from packet corruption.
#               Use only on physically secure networks.
#
auth 2
#1 crc
2 sha1 ultramonkey
#3 md5 Hello!
```

/etc/ha.d/haresources

```
#
#  /etc/ha.d/haresources
#
#  haresources to configure heartbeat with a master host mary.
#  Heartbeat will take over one IP address.
#
#  Based on sample haresources file shipped with heartbeat
#
#  Prepared: March 2003
#


#
#       This is a list of resources that move from machine to machine as
#       nodes go down and come up in the cluster.  Do not include
#       "administrative" or fixed IP addresses in this file.
#
#       We refer to this file when we're coming up, and when a machine is being
#       taken over after going down.
```

```
#
#       You need to make this right for your installation, then install it in
#       /etc/ha.d
#
#       These resources in this file are either IP addresses, or the name
#       of scripts to run to "start" or "stop" the given resource.
#
#       The format is like this:
#
#node-name resource1 resource2 ... resourceN
#
#       If the resource name contains an :: in the middle of it, the
#       part after the :: is passed to the resource script as an argument.
#   Multiple arguments are separated by the :: delimeter
#
#       In the case of IP addresses, the resource script name IPaddr is
#       implied.
#
#       For example, the IP address 135.9.8.7 could also be represented
#       as IPaddr::135.9.8.7
#
#       THIS IS IMPORTANT!!   vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
#
#       The given IP address is directed to an interface which has a route
#       to the given address.  This means you have to have a net route
#       set up outside of the High-Availability structure.  We don't set it
#       up here -- we key off of it.
#
#       The broadcast address for the IP alias that is created to support
#       an IP address defaults to the highest address on the subnet.
#
#       The netmask for the IP alias that is created defaults to the same
#       netmask as the route that it selected in in the step above.
#
```

```
#       The base interface for the IPalias that is created defaults to the
#       same netmask as the route that it selected in in the step above.
#
#       If you want to specify that this IP address is to be brought up
#       on a subnet with a netmask of 255.255.255.0, you would specify
#       this as IPaddr::135.9.8.7/8 .
#
#       If you wished to tell it that the broadcast address for this subnet
#       was 135.9.8.210, then you would specify that this way:
#                       IPaddr::135.9.8.7/8/135.9.8.210
#
#       If you wished to tell it that the interface to add the address to
#       is eth0, then you would need to specify it this way:
#                       IPaddr::135.9.8.7/8/eth0
#
# And this way to specify both the broadcast address and the
# interface:
#                       IPaddr::135.9.8.7/8/eth0/135.9.8.210
#
#       The IP addresses you list in this file are called "service" addresses,
#       since they're they're the publicly advertised addresses that clients
#       use to get at highly available services.
#
#       For a hot/standby (non load-sharing) 2-node system with only
#       a single service address,
#       you will probably only put one system name and one IP address in here.
#       The name you give the address to is the name of the default "hot"
#       system.
#
#       Where the nodename is the name of the node which "normally" owns the
#       resource.  If this machine is up, it will always have the resource
#       it is shown as owning.
#
#       The string you put in for nodename must match the uname -n name
```

```
#       of your machine.  Depending on how you have it administered, it could
#       be a short name or a FQDN.
#
#---------------------------------------------------------------
#
#       Simple case: One service address, default subnet and netmask
#               No servers that go up and down with the IP address
#
#just.linux-ha.org      135.9.216.110
#
#---------------------------------------------------------------
#
#       Assuming the adminstrative addresses are on the same subnet...
#       A little more complex case: One service address, default subnet
#       and netmask, and you want to start and stop http when you get
#       the IP address...
#
#just.linux-ha.org      135.9.216.110 http
#---------------------------------------------------------------
#
#       A little more complex case: Three service addresses, default subnet
#       and netmask, and you want to start and stop http when you get
#       the IP address...
#
#just.linux-ha.org      135.9.216.110 135.9.215.111 135.9.216.112 httpd
#---------------------------------------------------------------
#
#       One service address, with the subnet, interface and bcast addr
#  explicitly defined.
#
#just.linux-ha.org      135.9.216.3/4/eth0/135.9.216.12 httpd
#
#---------------------------------------------------------------
#
```

```
#       An example where a shared filesystem is to be used.
#       Note that multiple aguments are passed to this script using
#       the delimiter '::' to separate each argument.
#
#node1  10.0.0.170 Filesystem::/dev/sda1::/data1::ext2
```

mary 192.168.6.240/24/eth0

ตัวอย่างไฟล์คอนฟิกของการออกแบบในลักษณะ Load Balance

/etc/ha.d/ldirectord.cf

```
#
# Ldirectord will periodically connect to each real server
# and request a known URL. If the data returned by the server
# does not contain the the expected response then the
# test fails and the real server will be taken out of the available
# pool. The real server will be added back into the pool once the
# test succeeds. If all real servers are removed from the pool then
# localhost is added to the pool as a fallback measure.
#
# Based on the sample ldirectord.cf provided with ldirectord
#
# Prepared: March 2003
#

# Global Directives
checktimeout=10
checkinterval=2
#fallback=127.0.0.1:80
autoreload=no
#logfile="/var/log/ldirectord.log"
logfile="local0"
quiescent=yes
```

```
# Virtual Server for HTTP
virtual=10.1.3.61:80
        fallback=127.0.0.1:80
        real=192.168.6.4:80 masq
        real=192.168.6.5:80 masq
        service=http
        request="index.html"
        receive="Test Page"
        scheduler=rr
        #persistent=600
        protocol=tcp
                checktype=negotiate

# Virtual Service for HTTPS
virtual=10.1.3.61:443
        fallback=127.0.0.1:443
        real=192.168.6.4:443 masq
        real=192.168.6.5:443 masq
        service=https
        request="index.html"
        receive="Test Page"
        scheduler=rr
        #persistent=600
        protocol=tcp
                checktype=negotiate

# Virtual Service for FTP
# Note that peresistancy needs to be turned on for FTP when
# used with LVS-TUN (ipip) or LVS-DR (gate), but not with LVS-NAT (masq).
virtual=10.1.3.61:21
        fallback=127.0.0.1:21
        real=192.168.6.4:21 masq
        real=192.168.6.5:21 masq
        service=ftp
```

```
        request="welcome.msg"
        receive="Welcome"
                login="anonymous"
        passwd="anon@anon.anon"
        scheduler=rr
        #persistent=600
        protocol=tcp
                checktype=negotiate

## Virtual Service for IMAP
#virtual=10.1.3.61:143
#       fallback=127.0.0.1:143
#       real=192.168.6.4:143 masq
#       real=192.168.6.5:143 masq
#       service=imap
#               #login="test"
#       #passwd="test"
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
#       checktype=negotiate
#
## Virtual Service for POP
#virtual=10.1.3.61:110
#       fallback=127.0.0.1:110
#       real=192.168.6.4:110 masq
#       real=192.168.6.5:110 masq
#       service=pop
#               #login="test"
#       #passwd="test"
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
#
```

```
## Virtual Service for SMTP
#virtual=10.1.3.61:25
#       fallback=127.0.0.1:25
#       real=192.168.6.4:25 masq
#       real=192.168.6.5:25 masq
#       service=smtp
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
#
## Virtual Service for LDAP
#virtual=10.1.3.61:389
#       fallback=127.0.0.1:389
#       real=192.168.6.4:389 masq
#       real=192.168.6.5:389 masq
#       service=ldap
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
```

เราสามารถอ่านวิธีการคอนฟิก ldirectord โดยการใช้คำสั่ง ldirectord –help

ตัวอย่างไฟล์คอนฟิกของการออกแบบในลักษณะ High Availability and Load Balance

/etc/ha.d/ha.cf

```
# /etc/ha.d/ha.cf
#
# ha.cf file to configure two nodes connected by ethernet (eth0 and eth1) and
# a null modem (/dev/ttyS0).
#
# Based on sample ha.cf shipped with heartbeat
#
# Prepared: March 2003
#
```

```
#
#       There are lots of options in this file.  All you have to have is a set
#       of nodes listed {"node ...}
#       and one of {serial, bcast, mcast, or ucast}
#
#
#    Note on logging:
#    If any of debugfile, logfile and logfacility are defined then they
#    will be used. If debugfile and/or logfile are not defined and
#    logfacility is defined then the respective logging and debug
#    messages will be loged to syslog. If logfacility is not defined
#    then debugfile and logfile will be used to log messges. If
#    logfacility is not defined and debugfile and/or logfile are not
#    defined then defaults will be used for debugfile and logfile as
#    required and messages will be sent there.
#
#       File to write debug messages to
#debugfile /var/log/ha-debug
#
#
#       File to write other messages to
#
#logfile    /var/log/ha-log
#
#
#       Facility to use for syslog()/logger
#
logfacility  local0
#
#
#       A note on specifying "how long" times below...
#
#       The default time unit is seconds
#               10 means ten seconds
```

```
#
#       You can also specify them in milliseconds
#                       1500ms means 1.5 seconds
#
#
#       keepalive: how long between heartbeats?
#
keepalive 2
#
#       deadtime: how long-to-declare-host-dead?
#
deadtime 10
#
#       warntime: how long before issuing "late heartbeat" warning?
#       See the FAQ for how to use warntime to tune deadtime.
#
warntime 10
#
#
#       Very first dead time (initdead)
#
#       On some machines/OSes, etc. the network takes a while to come up
#       and start working right after you've been rebooted.  As a result
#       we have a separate dead time for when things first come up.
#       It should be at least twice the normal dead time.
#
initdead 10
#
#
#       nice_failback:  determines whether a resource will
#       fail back to its "primary" node, or remain on whatever
#       node is serving it until that node fails.
#
#       The default is "off", which means that it WILL fail
```

```
#       back to the node which is declared as primary in haresources
#
#       "on" means that resources only move to new nodes when
#       the nodes they are served on die.  This is deemed as a
#       "nice" behavior (unless you want to do active-active).
#
nice_failback on
#
#       hopfudge maximum hop count minus number of nodes in config
#hopfudge 1
#
#       serial  serialportname ...
serial  /dev/ttyS0  # Linux
#serial  /dev/cuaa0 # FreeBSD
#serial  /dev/cua/a  # Solaris
#
#
#       Baud rate for serial ports...
#
#baud    19200
#
#       What UDP port to use for communication?
#
#udpport    694
#
#       What interfaces to heartbeat over?
#
#bcast  eth0            # Linux
#bcast  le0             # Solaris
#
#       Set up a multicast heartbeat medium
#       mcast [dev] [mcast group] [port] [ttl] [loop]
#
#       [dev]                   device to send/rcv heartbeats on
```

```
#       [mcast group]       multicast group to join (class D multicast address
#                           224.0.0.0 - 239.255.255.255)
#       [port]              udp port to sendto/rcvfrom (no real reason to differ
#                           from the port used for broadcast heartbeats)
#       [ttl]               the ttl value for outbound heartbeats.  this effects
#                           how far the multicast packet will propagate.  (0-255)
#       [loop]              toggles loopback for outbound multicast heartbeats.
#                           if enabled, an outbound packet will be looped back and
#                           received by the interface it was sent on. (0 or 1)
#
#
mcast eth0 225.0.0.7 694 1 1
mcast eth1 225.0.0.7 694 1 1
#
#       Watchdog is the watchdog timer.  If our own heart doesn't beat for
#       a minute, then our machine will reboot.
#
#watchdog /dev/watchdog
#
#    "Legacy" STONITH support
#    Using this directive assumes that there is one stonith
#    device in the cluster.  Parameters to this device are
#    read from a configuration file.  The format of this line is:
#
#     stonith <stonith_type> <configfile>
#
#    NOTE: it is up to you to maintain this file on each node in the
#    cluster!
#
#stonith baytech /etc/ha.d/conf/stonith.baytech
#
#    STONITH support
#    You can configure multiple stonith devices using this directive.
#    The format of the line is:
```

```
#       stonith_host <hostfrom> <stonith_type> <params...>
#       <hostfrom> is the machine the stonith device is attached
#           to or * to mean it is accessible from any host.
#       <stonith_type> is the type of stonith device (a list of
#           supported drives is in /usr/lib/stonith.)
#       <params...> are driver specific parameters.  To see the
#           format for a particular device, run:
#        stonith -l -t <stonith_type>
#
#
#           Note that if you put your stonith device access information in
#           here, and you make this file publically readable, you're asking
#           for a denial of service attack ;-)
#
#
#stonith_host *     baytech 10.0.0.3 mylogin mysecretpassword
#stonith_host ken3  rps10 /dev/ttyS1 kathy 0
#stonith_host kathy rps10 /dev/ttyS1 ken3 0
#
#           Tell what machines are in the cluster
#           node        nodename ...        -- must match uname -n
node        fred
node        mary
#
#           Less common options...
#
#           Treats 10.10.10.254 as a psuedo-cluster-member
#
#ping 10.10.10.254
#
#           Started and stopped with heartbeat.  Restarted unless it exits
#                                               with rc=100
#
#respawn userid /path/name/to/run
```

/etc/ha.d/authkeys

```
#
# /etc/ha.d/authkeys
#
# Sample authkeys file. You should change the key and set the mode
# to 600
#
# Based on sample ha.cf shipped with heartbeat
#
# Prepared: July 2002
#
#       Authentication file.  Must be mode 600
#       Must have exactly one auth directive at the front.
#       auth    send authentication using this method-id
#       Then, list the method and key that go with that method-id
#       Available methods: crc sha1, md5.  Crc doesn't need/want a key.
#       You normally only have one authentication method-id listed in this file
#       Put more than one to make a smooth transition when changing auth
#       methods and/or keys.

#       sha1 is believed to be the "best", md5 next best.
#
#       crc adds no security, except from packet corruption.
#               Use only on physically secure networks.
#
auth 2
#1 crc
2 sha1 ultramonkey
#3 md5 Hello!
```

/etc/ha.d/haresources

```
#
#       /etc/ha.d/haresources
#
#       haresources to configure heartbeat with a master host mary.
#       Heartbeat will take-over two IP addresses and run
#       ldirectord to monitor the real servers.
#
#       Based on sample haresources file shipped with heartbeat
#
#       Prepared: March 2003
#

#
#               This is a list of resources that move from machine to machine as
#               nodes go down and come up in the cluster.  Do not include
#               "administrative" or fixed IP addresses in this file.
#
#               We refer to this file when we're coming up, and when a machine is being
#               taken over after going down.
#
#               You need to make this right for your installation, then install it in
#               /etc/ha.d
#
#               These resources in this file are either IP addresses, or the name
#               of scripts to run to "start" or "stop" the given resource.
#
#               The format is like this:
#
#node-name resource1 resource2 ... resourceN
#
#               If the resource name contains an :: in the middle of it, the
#               part after the :: is passed to the resource script as an argument.
```

```
#	Multiple arguments are separated by the :: delimeter
#
#		In the case of IP addresses, the resource script name IPaddr is
#		implied.
#
#		For example, the IP address 135.9.8.7 could also be represented
#		as IPaddr::135.9.8.7
#
#		THIS IS IMPORTANT!!	vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
#
#		The given IP address is directed to an interface which has a route
#		to the given address.  This means you have to have a net route
#		set up outside of the High-Availability structure.  We don't set it
#		up here -- we key off of it.
#
#		The broadcast address for the IP alias that is created to support
#		an IP address defaults to the highest address on the subnet.
#
#		The netmask for the IP alias that is created defaults to the same
#		netmask as the route that it selected in in the step above.
#
#		The base interface for the IPalias that is created defaults to the
#		same netmask as the route that it selected in in the step above.
#
#		If you want to specify that this IP address is to be brought up
#		on a subnet with a netmask of 255.255.255.0, you would specify
#		this as IPaddr::135.9.8.7/8 .
#
#		If you wished to tell it that the broadcast address for this subnet
#		was 135.9.8.210, then you would specify that this way:
#			IPaddr::135.9.8.7/8/135.9.8.210
#
#		If you wished to tell it that the interface to add the address to
#		is eth0, then you would need to specify it this way:
```

```
#                       IPaddr::135.9.8.7/8/eth0
#
#       And this way to specify both the broadcast address and the
#       interface:
#                       IPaddr::135.9.8.7/8/eth0/135.9.8.210
#
#               The IP addresses you list in this file are called "service" addresses,
#               since they're they're the publicly advertised addresses that clients
#               use to get at highly available services.
#
#               For a hot/standby (non load-sharing) 2-node system with only
#               a single service address,
#               you will probably only put one system name and one IP address in here.
#               The name you give the address to is the name of the default "hot"
#               system.
#
#               Where the nodename is the name of the node which "normally" owns the
#               resource.  If this machine is up, it will always have the resource
#               it is shown as owning.
#
#               The string you put in for nodename must match the uname -n name
#               of your machine.  Depending on how you have it administered, it could
#               be a short name or a FQDN.
#
#-------------------------------------------------------------------
#
#               Simple case: One service address, default subnet and netmask
#                       No servers that go up and down with the IP address
#
#just.linux-ha.org      135.9.216.110
#
#-------------------------------------------------------------------
#
#               Assuming the adminstrative addresses are on the same subnet...
```

```
#       A little more complex case: One service address, default subnet
#       and netmask, and you want to start and stop http when you get
#       the IP address...
#
#just.linux-ha.org      135.9.216.110 http
#-------------------------------------------------------------------
#
#       A little more complex case: Three service addresses, default subnet
#       and netmask, and you want to start and stop http when you get
#       the IP address...
#
#just.linux-ha.org      135.9.216.110 135.9.215.111 135.9.216.112 httpd
#-------------------------------------------------------------------
#
#       One service address, with the subnet, interface and bcast addr
#   explicitly defined.
#
#just.linux-ha.org      135.9.216.3/4/eth0/135.9.216.12 httpd
#
#-------------------------------------------------------------------
#
#   An example where a shared filesystem is to be used.
#   Note that multiple aguments are passed to this script using
#   the delimiter '::' to separate each argument.
#
#node1  10.0.0.170 Filesystem::/dev/sda1::/data1::ext2

mary IPaddr::192.168.6.240/24/eth0 IPaddr::192.168.7.240/24/eth1 ldirectord::ldirectord.cf
```

/etc/ha.d/ldirectord

```
#
# Ldirectord will periodically connect to each real server
# and request a known URL. If the data returned by the server
# does not contain the the expected response then the
# test fails and the real server will be taken out of the available
# pool. The real server will be added back into the pool once the
# test succeeds. If all real servers are removed from the pool then
# localhost is added to the pool as a fallback measure.
#
# Based on the sample ldirectord.cf provided with ldirectord
#
# Prepared: March 2003
#

# Global Directives
checktimeout=10
checkinterval=2
#fallback=127.0.0.1:80
autoreload=no
#logfile="/var/log/ldirectord.log"
logfile="local0"
quiescent=yes

# Virtual Server for HTTP
virtual=192.168.7.240:80
        fallback=127.0.0.1:80
        real=192.168.6.4:80 masq
        real=192.168.6.5:80 masq
        service=http
        request="index.html"
        receive="Test Page"
        scheduler=rr
```

```
        #persistent=600
        protocol=tcp
                checktype=negotiate

# Virtual Service for HTTPS
virtual=192.168.7.240:443
        fallback=127.0.0.1:443
        real=192.168.6.4:443 masq
        real=192.168.6.5:443 masq
        service=https
        request="index.html"
        receive="Test Page"
        scheduler=rr
        #persistent=600
        protocol=tcp
                checktype=negotiate

# Virtual Service for FTP
# Note that peresistancy needs to be turned on for FTP when
# used with LVS-TUN (ipip) or LVS-DR (gate), but not with LVS-NAT (masq).
virtual=192.168.7.240:21
        fallback=127.0.0.1:21
        real=192.168.6.4:21 masq
        real=192.168.6.5:21 masq
        service=ftp
        request="welcome.msg"
        receive="Welcome"
                login="anonymous"
        passwd="anon@anon.anon"
        scheduler=rr
        #persistent=600
        protocol=tcp
                checktype=negotiate
```

```
## Virtual Service for IMAP
#virtual=192.168.7.240:143
#       fallback=127.0.0.1:143
#       real=192.168.6.4:143 masq
#       real=192.168.6.5:143 masq
#       service=imap
#       #login="test"
#       #passwd="test"
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
#       checktype=negotiate
#
## Virtual Service for POP
#virtual=192.168.7.240:110
#       fallback=127.0.0.1:110
#       real=192.168.6.4:110 masq
#       real=192.168.6.5:110 masq
#       service=pop
#       #login="test"
#       #passwd="test"
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
#
## Virtual Service for SMTP
#virtual=192.168.7.240:25
#       fallback=127.0.0.1:25
#       real=192.168.6.4:25 masq
#       real=192.168.6.5:25 masq
#       service=smtp
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
```

```
#       checktype=negotiate
#
## Virtual Service for LDAP
#virtual=192.168.7.240:389
#       fallback=127.0.0.1:389
#       real=192.168.6.4:389 masq
#       real=192.168.6.5:389 masq
#       service=ldap
#       scheduler=rr
#       #persistent=600
#       protocol=tcp
#       checktype=negotiate
#
```

ภาคผนวก ข วิธีการติดตั้งโปรแกรมที่ใช้ในงานวิจัยบนลีนุกซ์

การติดตั้งเว็บเซิร์ฟเวอร์ Apache

ซอฟต์แวร์เว็บเซิร์ฟเวอร์ในปัจจุบันมีออกมามากมาย แต่ที่เป็นที่นิยมกันก็คงเป็นของค่ายอาปาเช่ น (Apache) ไม่ว่าจะใช้เป็นเว็บเซิร์ฟเวอร์บนลีนุกซ์ ยูนิกซ์ หรือระบบปฏิบัติการ Windows ก็ตาม

สำหรับลีนุกซ์ค่ายต่าง ๆ ก็ได้นำ Apache เว็บเซิร์ฟเวอร์รวมมากับแผ่นลีนุกซ์ ซึ่งจะทำการติดตั้ง Apache เว็บเซิร์ฟเวอร์ให้ด้วยสำหรับการติดตั้งแบบปกติ

ดังนั้นในบทนี้เราจะมาทำความรู้จักกับ Apache เว็บเซิร์ฟเวอร์กัน ตั้งแต่การติดตั้งจนถึงการใช้งานและปรับแต่งเว็บเซิร์ฟเวอร์ให้เหมาะกับการใช้งาน

รู้จักกับ Apache

Apache เว็บเซิร์ฟเวอร์เริ่มต้นเกิดจากการนำซอฟต์แวร์ NCSA HTTP 1.3 Server มาพัฒนาโดยกลุ่มผู้พัฒนาได้รวมตัวกันในโครงการ "Apache Software Foundation" ซึ่งจุดมุ่งหมายเพื่อจัดทำเว็บเซิร์ฟเวอร์ที่เป็นมาตรฐานขึ้น โดยไม่ปกปิดโค้ดต้นฉบับ

ในปัจจุบัน Apache เว็บเซิร์ฟเวอร์ได้ออกเวอร์ชั่น 2.0.x แล้ว โดยเวอร์ชันก่อนหน้ายังเป็น 1.3.xx สำหรับซอฟต์แวร์ หรือรายละเอียดเกี่ยวกับคุณสมบัติต่าง ๆ สามารถดาวน์โหลดได้ที่ http://www.apache.org และส่วนของคู่มือเราอาจเข้าไปอ่านได้จากไดเร็คทอรี docs/manual ซึ่งจะอยู่ในไฟล์โค้ดต้นฉบับที่เราดาวน์โหลดมานั่นเอง

เริ่มติดตั้งซอฟต์แวร์เว็บเซิร์ฟเวอร์

การติดตั้งแบ่งออกเป็น 2 แบบด้วยกัน ขึ้นอยู่กับซอฟต์แวร์โปรแกรมที่เรานำมาติดตั้ง นั่นคือเป็นแบบแพ็คเกจ (.rpm) หรือเป็นแบบโค้ดต้นฉบับ (.tar.gz) ซึ่งวิธีการติดตั้งก็ไม่แตกต่างจากการติดตั้งเซิร์ฟเวอร์แบบอื่น ๆ นัก โดยมีขั้นตอนการติดตั้งดังนี้

ติดตั้งซอฟต์แวร์ Apache เว็บเซิร์ฟเวอร์แบบแพ็คเกจ

สำหรับซอฟต์แวร์แพ็คเกจ (.rpm) เราจะเรียกโปรแกรม rpm ช่วยในการติดตั้ง ดังนี้

[root @ redhat root] # rpm –iv apacha-1.3.22-1.7.1.i386.rpm

ในแต่ละเวอร์ชันก็อาจมีความต้องการโมดูลไม่เท่ากัน ซึ่งบางทีตอนติดตั้งอาจพบว่ามีข้อความแจ้งว่าซอฟต์แวร์เวอร์ชันนี้ต้องการโมดูลตัวนี้เพิ่ม ซึ่งเราต้องไปดาวน์โหลดมาลงเพิ่มด้วย

ติดตั้งซอฟต์แวร์ Apache เว็บเซิร์ฟเวอร์แบบโค้ดต้นฉบับ

สำหรับซอฟต์แวร์ที่มาแบบโค้ดต้นฉบับ เราต้องคอมไพล์ก่อนการติดตั้ง ซึ่งขั้นตอนอาจมากกว่าติดตั้งแบบแพ็คเกจ แต่ก็ไม่ยุ่งยากนัก โดยทั่วไปจะไม่ไฟล์ที่บอกขั้นตอนการติดตั้งมาด้วย สำหรับ Apache เว็บเซิร์ฟเวอร์ หลังจากแตกไฟล์ tar แล้ว ควรอ่านไฟล์ README ก่อน ส่วนขั้นตอนการติดตั้งอ่านในไฟล์ INSTALL ซึ่งจะมีขั้นตอนดังต่อไปนี้

สิ่งที่ต้องเตรียมก่อนการติดตั้ง

สำหรับสิ่งต่าง ๆ ที่ต้องเตรียมให้พร้อมมีดังนี้

1. **เนื้อที่ว่างในฮาร์ดดิสก์ :** สำหรับเวอร์ชัน 1.3.xx อย่างน้อยต้องมีเนื้อที่ 12 MB ไว้ใช้ในการติดตั้งชั่วคราว หลังจากติดตั้งเสร็จแล้วจะใช้เนื้อที่ประมาณ 6 MB ถ้าเป็นเวอร์ชัน 2.x ต้องมีอย่างน้อย 50 MB ไว้ใช้ในการติดตั้งชั่วคราว หลังจากติดตั้งแล้วใช้เนื้อที่ประมาณ 13 MB
2. **ANSI-C Compiler :** เตรียม C คอมไพเลอร์ ซึ่งเราใช้ของ GNU C compiler (gcc) ก็ได้เป็นฟรีซอฟต์แวร์ ดาวน์โหลดได้ที่ http://www.gnu.org แนะนำให้ใช้เวอร์ชัน 2.7.2 โดยปกติบนลีนุกซ์จะเตรียมมาให้แล้ว
3. **Perl 5 Interpreter :** สำหรับ Perl ในขั้นตอนการคอมไพล์จะมีเรียกใช้ ถ้าไม่มีก็ยังคอมไพล์ได้แต่ก็จะไม่มีการสนับสนุนการใช้งาน Perl ไปด้วย แต่โดยปกติลีนุกซ์ได้เตรียมไว้ให้แล้ว
4. โปรแกรม **make** : ให้ตรวจสอบว่ามีการกำหนดพาธถึงหรือยัง โดยปกติอยู่ที่ไดเร็คทอรี /usr/bin/
5. ทั้ง C คอมไพเลอร์ และ Perl เราต้องกำหนดพาธให้ถึงด้วย

แตก **tar** ไฟล์โค้ดต้นฉบับ **[root @ redhat tmp] # tar xvfz apache_1.3.24.tar.gz**

คอมไพล์และติดตั้ง

เปลี่ยนไดเร็คทอรีเข้าไปยังไดเร็คทอรีที่แตกจากไฟล์ tar มา แล้วเรียกคำสั่งดังต่อไปนี้

[root @ redhat apache_1.3.24] # ./configure --prefix= ไดเร็คทอรีที่ต้องการให้ติดตั้ง โปรแกรม

เช่น

 # ./configure --prefix=/opt/Apache 1-3-24

[root @ redhat apache_1.3.24] # make

[root @ redhat apache_1.3.24] # make install

หลังจากเสร็จการเรียกคำสั่งทั้ง 3 แล้ว จะมีไดเร็คทอรีตามที่กำหนดที่ prefix ซึ่งจะกล่าวถึงไฟล์ต่าง ๆ ในหัวข้อถัดไป

ทดลองหลังการติดตั้ง

เมื่อติดตั้งซอฟต์แวร์เว็บเซิร์ฟเวอร์เสร็จแล้ว ก็ลองทดสอบดูว่าเราติดตั้งสมบูรณ์หรือไม่ โดยการเรียกคำสั่งให้เว็บเซิร์ฟเวอร์ทำงาน แล้วใช้บราวเซอร์เข้าเว็บทดสอบดู ดังนี้

จากตัวอย่างได้ติดตั้งซอฟต์แวร์ไว้ที่ /opt/Apache1-3-24 ดังนั้นให้เรียกคำสั่งดังนี้

[root @ redhat root] # /opt/Apache1 – 3 – 24/bin/apachectl start

/opt/Apache1-3-24/bin/apachectl start : httpd started.

จากตัวอย่างพบว่ามีเดมอน นั่นหมายความว่าเว็บเซิร์ฟเวอร์ทำงานแล้ว คราวนี้ก็มาทดลองเข้าเว็บผ่านเว็บเซิร์ฟเวอร์ตัวนี้กัน ผ่านเว็บเซิร์ฟเวอร์เตรียมไว้ให้สำหรับทดสอบ เพียงแต่ใส่ URL เป็น http://localhost

ถ้าปรากฏเว็บเพจทดสอบขึ้นมา แสดงว่าเราติดตั้งเสร็จสมบูรณ์แล้ว คราวนี้ก็มาปรับค่าให้สามารถใช้งานกับเว็บเพจของเรากันต่อ

รู้จักไฟล์ปรับค่า (Configuration File)

ไฟล์ปรับค่าของ Apache เว็บเซิร์ฟเวอร์ ถ้าติดตั้งแบบแพ็คเกจไดเร็คทอรีที่เก็บไฟล์ปรับค่าจะอยู่ที่ /etc/httpd/conf แต่ถ้าเราติดตั้งแบบโค้ดต้นฉบับ ไดเร็คทอรีที่เก็บไฟล์ปรับค่าจะอยู่ที่ *prefix*/conf (prefix = ไดเร็คทอรีที่กำหนดไว้ติดตั้งโปรแกรม ในที่นี้คือ /opt/Apache1-3-24) ภายใต้ไดเร็คทอรีนี้ประกอบไปด้วยไฟล์ต่าง ๆ ดังนี้

http.conf : เป็นไฟล์หลักในการปรับแต่งเว็บเซิร์ฟเวอร์ ซึ่งภายในประกอบไปด้วย Directive ต่าง ๆ ให้เรากำหนดค่าเพื่อให้เว็บเซิร์ฟเวอร์เป็นไปอย่างที่เราต้องการได้ เช่น กำหนดไดเร็คทอรีที่เก็บเว็บเพจ กำหนดพอร์ตที่เข้ามาใช้เว็บ เป็นต้น ในรายละเอียดจะกล่าวถึงในหัวข้อถัดไป

arm.conf : เป็นไฟล์ที่เก็บข้อมูลเกี่ยวกับไดเร็คทอรี ไฟล์ และชนิดของไฟล์

access.conf : เป็นไฟล์ที่ใช้ควบคุมการเข้ามาใช้งานเว็บเซิร์ฟเวอร์ของเรา ซึ่งจะถูกเรียกใช้เมื่อเรากำหนดในไฟล์ httpd.conf เปิดให้ directive ที่ชื่อ "AccessConfig conf/access.conf" ใช้งาน ไฟล์นี้จะถูกเรียกอ่านหลังจากไฟล์ srm.conf ซึ่งมีรูปแบบดังตัวอย่างนี้

ตัวอย่างที่ 1 : ไม่เปิดให้เครื่อง bomber.unknow.com เข้ามาใช้งาน แต่เครื่องอื่นให้เข้ามาได้

```
<Directory "...">
< Limit GET>
order allow , deny          (ลำดับในการตรวจสอบในที่นี้คืออ่าน deny ก่อน allow)
allow from all              (ยอมให้เข้ามาใช้เว็บเซิร์ฟเวอร์ได้ทุกคน)
deny from bomber.unknow.com
< Limit >                   (ไม่ยอมให้เครื่อง bomber.unknow.com เข้ามาใช้งาน)
</Directory>
```

จากตัวอย่างที่ 1 ลำดับในการเข้ากฎคือ allow ทุกเครื่องให้เข้ามาใช้งานเซิร์ฟเวอร์ได้ แล้วจึง deny เครื่อง bomber.unknow.com

ตัวอย่างที่ 2 : เปิดให้เครื่อง goodguy.youknow.com เข้ามาใช้งานเครื่องเดียว

```
<Directory "...">           (ลำดับในการตรวจสอบในที่นี้คืออ่าน deny ก่อน allow)
< Limit GET >
order deny,allow            (ยอมให้เข้ามาใช้เฉพาะเครื่อง goodguy.youknow.com)
allow from goodguy.youknow.com
deny from all               (ไม่ยอมให้เครื่องไหนเข้ามาเลย)
< Limit >
</Directory>
```

จากตัวอย่างที่ 2 ลำดับในการเข้ากฎคือ deny เครื่องทุกเครื่องก่อน หลังจากนั้นจึง allow เครื่อง goodguy.youknow.com เพราะฉะนั้นก็จะมีเพียงเครื่องเดียวที่เข้ามาใช้งานได้

mime.type : เก็บรูปแบบของ mime ที่ใช้ในการรับส่งข้อมูลประเภทต่าง ๆ ไม่ควรเปลี่ยนแปลงข้อความในไฟล์

magic : เก็บรูปแบบข้อมูลสำหรับโมดูล mod_mime_magic ไม่ควรเปลี่ยนแปลงข้อความในไฟล์

รู้จักกับไฟล์ httpd.conf

สำหรับการปรับแต่ง Apache เว็บเซิร์ฟเวอร์ ไฟล์ httpd.conf ถือเป็นไฟล์หลักที่เราต้องนึกถึง โดยที่ออปชันที่จะกำหนดหรือปรับค่าเราเรียกว่า ไดเร็คทีฟ (Directive) ซึ่งสามารถแบ่งไดเร็คทีฟออกเป็นส่วน ๆ ตามความหมายได้ดังหัวข้อต่อไปนี้

ส่วนที่ 1 : สภาพแวดล้อมโดยรวม (Global Environment)

การกำหนดค่า Directive ในส่วนนี้ ส่งผลถึงการทำงานโดยรวมของ Apache ซึ่งจะมี Directive ดังตารางที่ 1

Directive	คำอธิบาย
ServerType	กำหนดรูปแบบการทำงานของเซิร์ฟเวอร์เป็น standalone หรือ inetd ค่าเริ่มต้นจะกำหนดเป็น satndalone
ServerRoot	ระบุไดเร็คทอรีที่เก็บไฟล์ Configuration, ไฟล์ Log. ไฟล์ Error โดยปกติจะกำหนดเป็นไดเร็คทอรีที่เราติดตั้งโปรแกรม (ตามค่า prefix) เช่น ServerRoot "/opt/Apache1-3-24"
LockFile	ระบุตำแหน่งและชื่อไฟล์ที่ต้องการใช้เป็นล็อคไฟล์ เช่น LockFile /opt/Apache1-3-24/logs/httpd.lock
PidFile	ระบุตำแหน่งและชื่อไฟล์ที่ใช้เก็บหมายเลขโพรเซสของเว็บเซิร์ฟเวอร์ทุกครั้งที่เริ่มทำงาน เช่น PidFile/opt/Apache1-3-24*logs/httpd.pid
ScoreBoardFile	ระบุไฟล์ที่ใช้เก็บหมายเลขของเซิร์ฟเวอร์โพรเซส (บนลีนุกซ์ไม่ใช้ได้เร็คทีฟนี้ป
ResourceConfig	กำหนดให้มีการเรียกใช้ไฟล์ srm.conf เช่น ResourceConfig conf/srm.conf
AccessConfig	กำหนดให้มีการเรียกใช้ไฟล์ access.conf เช่น ResourceConfig conf/access.conf
Timeout	ระบุช่วงเวลาในการรับ/ส่งข้อมูล หน่วยเป็นวินาที เช่น Timeout 300
KeepAlive	กำหนดว่าจะยอมให้มีการร้องขอได้มากกว่าหนึ่งในแต่ละการติดต่อ โดยกำหนดด้วยค่า On/Off เช่น KeepAlive On
MaxKeepAliveRequests	ระบุจำนวนการร้องขอมากที่สุดในแต่ละการติดต่อ ถ้าไม่จำกัดจำนวนให้กำหนดเป็นค่า 0 แนะนำให้กำหนดเป็นค่าจำนวนหนึ่งที่มาก ๆ ดีกว่า เช่น MaxKeepAliveRequests 100
KeepAliveTimeout	ระบุช่วงเวลาในการรอการร้องขอครั้งต่อไปในแต่ละการติดต่อ เช่น KeepAliveTimeout 15
MinSpareServers	ระบุจำนวนเซิร์ฟเวอร์โพรเซสน้อยที่สุดที่ถูกเรียกขึ้น เช่น MinSpareServers 5
MaxSpareServers	ระบุจำนวนเซิร์ฟเวอร์โพรเซสมากที่สุดที่ระบบจะเรียกขึ้นมาได้ เช่น MaxSpareServers 10

StartServers	ระบุจำนวนเซิร์ฟเวอร์โพรเซสเริ่มต้น ตรวจสอบได้จากคำสั่ง “ps -ef | grep http” (ไม่นับเซิร์ฟเวอร์ที่มีโพรเซสพ่อเป็น 1) เช่น StartServers 4
MaxClient	ระบุจำนวนเครื่องไคลเอนท์สูงสุดที่เข้ามาติดต่อได้ เช่น MaxClients 200
MaxRequestsPerChild	ระบุจำนวนการร้องขอที่โพรเซสลูกรองรับได้ ก่อนที่โพรเซสลูกจะตาย เช่น MaxRequestsPerChild 0
Listen	กำหนดพอร์ตหรือระบุเป็น IP : Port ที่ใช้สำหรับรับการร้องขอจากไคลเอนท์ เช่น Listen3000Listen 203.10.103.22 : 3000
BindAddress	สนับสนุนการใช้เวอร์ชวลโฮสต์ (Virtual hosts) เราสามารถระบุได้ทั้ง " * ", " IP " หรือ “ชื่อเต็มที่ระบุโดเมนเนม” เช่น BindAddress
ExtendedStatus	กำหนดเป็น On แสดงข้อมูลสถานะแบบเต็ม กำหนดเป็น Off แสดงข้อมูลสถานะแบบปกติเมื่อมีการเรียกใช้ “server-status” เช่น ExtendedStatus On

ตารางที่ 1 (ต่อ) : แสดง Directive ในส่วนสภาพแวดล้อมโดยรวม

ส่วนที่ 2 : **Dynamic Shared Object (DSO) Support**

ในส่วนนี้เป็นการกำหนดการโหลดโมดูลที่ต้องการใช้ให้เข้ากันกับไดเร็คทีฟที่เรากำหนด โดยมีรูปแบบในการกำหนด พร้อมทั้งตัวอย่างดังนี้

LoadModule ชื่อโมดูล ไฟล์โมดูลนั้น

```
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding LoadModule' lines at this location so the
# directives contained in it are actually available _before_they are used.
# Please read the file http://httpd.apache.org/docs/dos.html for more
# details about the DSO mechanism and run 'httpd -l' for the list of already
# built-in (statically linked and thus always available) modules in your httpd
# binary.
#
# Note: The order in which modules are loaded is important. Don't change
# the order below without expert advice.
#
# Example:
```

```
# LoadModule foo_module libexec/mod_foo.so
LoadModule env_module    modules/mod_env.so
LoadModule mime_module   modules/mod_mime.so
LoadModule info_module   modules/mod_info.so
```

ส่วนที่ 3 : เซิร์ฟเวอร์ไดเร็คทีฟ (Server Configuration)

ในส่วนนี้เป็นการกำหนดค่าที่เซิร์ฟเวอร์ต้องใช้ ซึ่งไดเร็คทีฟต่าง ๆ ในส่วนนี้สามารถนำไปใช้กับการกำหนดเวอร์ชวลโฮสต์ <VirtualHost> ได้

Directive	คำอธิบาย
Port	กำหนดพอร์ตเพื่อรับการร้องขอจากไคลเอนท์ สำหรับเซิร์ฟเวอร์แบบ standalone ถ้ากำหนดหมายเลขพอร์ต < 1023 เดมอน httpd ต้องถูกเรียกขึ้นมาด้วยผู้ใช้ root เช่น Port 80
User/Group	กำหนดผู้ใช้และกลุ่มผู้ใช้ (uid/gid) สามารถกำหนดเป็นชื่อหรือหมายเลขก็ได้ สำหรับใช้ในการเรียก httpd ทำงาน เช่น User nobodyGroup nobody
ServerAdmin	ระบุ email address ผู้รับเมล เมื่อเซิร์ฟเวอร์มีปัญหา เช่น ServerAdmin webmaster@zoo.co.th
ServerName	ระบุโฮสต์เนมเครื่องเว็บเซิร์ฟเวอร์ ถ้าต้องการให้ภายนอกติดต่อได้ เราต้องให้เซิร์ฟเวอร์เนมนี้อยู่ใน DNS ด้วย เช่น ServerName redhat.zoo.co.th
DocumentRoot	กำหนดไดเร็คทอรีที่ให้เซิร์ฟเวอร์เข้ามาอ่านเว็บเพจ หรือข้อมูลที่ต้องการให้บริการ เช่น DocumentRoot "/www/docs-web"
UserDir	กำหนดชื่อไดเร็คทอรีที่ผู้ใช้แต่ละคนสามารถสร้างขึ้นที่โฮมของตัวเอง เพื่อแสดงเว็บเพจของแต่ละคนโดยนำไฟล์ ".html" มาใส่ไว้ UserDir public_html
DirectoryIndex	กำหนดชื่อไฟล์ html ให้เซิร์ฟเวอร์เรียกขึ้นมาโดยอัตโนมัติ โดยไม่ต้องระบุชื่อไฟล์ html เช่น <lfModule mod_dir.c> Eirvctoryindex index.html main.html index.htm</lfModule>
AccessFileName	กำหนดไฟล์ที่ใช้ควบคุมการเข้าถึงไดเร็คทอรีที่ไฟล์นี้อยู่ โดยปกติกำหนดเป็นชื่อ ".htaccess" เช่น AccessFileName. Htaccess
CacheNegotiatedDocs	กำหนดให้มีการเก็บข้อมูลไว้ชั่วคราว (caching) สำหรับเอกสารที่เคยอ่านไว้ การกำหนดให้ใช้เพียงเอาเครื่องหมาย "#" ข้างหน้าออกเท่านั้น

UseCanonicalName	ในกรณีที่เราต้องการสร้าง self-referencing URL เราสามารถกำหนด Canonical ให้เป็นชื่อเซิร์ฟเวอร์ : พอร์ต แล้วเรียกผ่าน Canonical ได้เลย แต่ถ้าเราไม่เปิดใช้ไดเร็คทีฟนี้เราก็ต้องกำหนดเป็นชื่อเซิร์ฟเวอร์ : พอร์ต ตรง ๆ เลย ค่าโดยปกติเป็น UseCanonicalName On
TypesConfig	ระบุไดเร็คทอรีที่อยู่พร้อมชื่อไฟล์ mime.types เช่น <lfModule mod_mime.c> TypesConfig/opt/Apache1-3-24/conf/mime.types</lfModule>
DefaultType	กำหนดชนิดของ mime ที่เซิร์ฟเวอร์จะใช้ในเอกสาร ถ้าเซิร์ฟเวอร์เราเก็บเอกสารส่วนใหญ่เป็นไฟล์ข้อความหรือไฟล์ HTML ควรกำหนดเป็น "text/plain" แต่ถ้าเอกสารส่วนใหญ่เป็นพวกไบนารีเราควรกำหนดเป็น "application/xxx" แทน เช่น DefaultType text/plain
MIMEMagicFile	ระบุไดเร็คทอรีที่อยู่พร้อมชื่อไฟล์ magic เช่น <lfModule mod_mime_magic.c> MIMEMagicFile/opt/Apache1-3-24/conf/magic</lfModule>
HostnameLookups	บันทึกชื่อของไคลเอนท์ถ้ากำหนดเป็น "On" และจะบันทึกเป็น IP ถ้ากำหนดเป็น "Off" เช่น HostnameLookups Off
ErrorLog	กำหนดไดเร็คทอรีและไฟล์ที่ใช้บันทึกข้อผิดพลาด ในกรณีที่ใช้เวอร์ชวลโฮสต์ ถ้าไม่มีการกำหนด ErrorLog ของเวอร์ชวลโฮสต์นั้น ๆ ไว้ ข้อผิดพลาดที่เกิดขึ้นจะเก็บรวมไว้ที่ไฟล์ที่กำหนดตรงนี้ เช่น ErrorLog/opt/Apache1-3-24/logs/error_log
LogLevel	กำหนดระดับของการบันทึกข้อความที่เกิดขึ้นกับเซิร์ฟเวอร์ แบ่งได้ดังนี้ debug, info, motice, warn, error, crit, alert, emerg เช่น LogLevel emerg
LogFormat	รูปแบบในการบันทึกลงล็อกไฟล์ สำหรับ CustomLog เช่น LogFormat "%h %l %u %t \ "%r\" %>s %b" commonLogFormat "%{User-agent}I" agent
CustomLog	ระบุตำแหน่งและไฟล์ที่ใช้บันทึกการ access ล็อกไฟล์ ถ้าเราใช้เวอร์ชวลโฮสต์ แต่ไม่ได้กำหนด CustomLog ในแต่ละเวอร์ชวลโฮสต์นั้น ๆ การบันทึกลงล็อกจะทำที่ไฟล์ที่กำหนด ณ ที่นี้แทน เช่น CustomLog/opt/Apache1-3-24/logs/access_log commonCustomLog /opt/Apache1-3-24/logs/agent_log agent
MetaDir	ระบุไดเร็คทอรีที่เก็บ meta information ไฟล์ ภายในไฟล์เก็บ HTTP Header ไว้ เช่น MetaDir .web

MetaSuffix	กำหนดชื่อของ meta information ไฟล์ เช่น MetaSuffix .meta
Directive	คำอธิบาย
ErrorDocument	ปรับแต่งข้อความ error แบ่งเป็น 3 แบบ q ข้อความ ErrorCocument 500 "The server missing something and still don't know" q เรียกจากไฟล์บนเซิร์ฟเวอร์ ErrorCocument 404 /missing.html หรือ ErrorDocument 404 /cgi-bin/missing_handler.plq เรียกจากไฟล์ต่างเซิร์ฟเวอร์ EroorDocument 402 http://unknow.other.com/moticesomething.html
BrowserMatch	กำหนด keepalives และ HTTP header เช่น เรากำหนดไม่ให้มีการใช้โปรโตรคอล HTTP/1.1 สำหรับบราวเซอร์ที่ไม่สนับสนุนการใช้ เช่น Netscape 2.x และ IE 4.0 BrowserMatch "Mozilla/2" mokeepaliveBrowserMatch "MSIE 4/.0b2." mokeepalive downgrade 1.0 force-response-1.0
ProxyRequest	เปิดให้ใช้พร็อกซี่เซิร์ฟเวอร์

ตารางที่ 2 : แสดง Directive ในส่วนของเซิร์ฟเวอร์

การกำหนดคุณสมบัติของไดเร็คทอรี

ในแต่ละไดเร็คทอรีที่ Apache เว็บเซิร์ฟเวอร์เข้าไปใช้ เราสามารถกำหนดคุณสมบัติ สิทธิ์ในการเข้าไปใช้ไดเร็คทอรีต่าง ๆ นั้นได้ ซึ่งภายในไดเร็คทอรีก็มีไดเร็คทีฟ ดังตารางที่ 3

Directive	คำอธิบาย
Options	ส่วนที่กำหนดเกี่ยวกับคุณสมบัติของไดเร็คทอรีประกอบด้วย **Indexes** ในกรณีที่ไคลเอนท์มีการเข้า URL มายังไดเร็คทอรีที่เซิร์ฟเวอร์กำหนดไว้ และภายในไดเร็คทอรีนี้ ไม่มีไฟล์ที่กำหนดไว้ใน **DirectoryIndex เช่น index.html** เป็นต้น ก็ให้เซิร์ฟเวอร์แสดงเป็นรายชื่อไฟล์ที่อยู่ภายในไดเร็คทอรีนี้แทน Includes กำหนดให้ Server-side includes (SSI) ทำงาน FollowSymLinks เซิร์ฟเวอร์จะ follow symbolic links ในไดเร็คทอรีนี้ ExecCGI กำหนดให้สามารถเรียกใช้งานสคริปต์ CGI ได้
AllowOverride	เมื่อเซิร์ฟเวอร์พบว่ามีไฟล์ .htaccess อยู่ที่ไดเร็คทอรีที่กำหนดนี้ ไดเร็คทีฟนี้จะระบุออปชันควบคุมไฟล์นี้ AuthConfig ใช้เพิ่มความปลอดภัย โดยให้มีการล็อกอินเข้ามา เมื่อมีการใช้ไดเร็คทอรีนี้ FileInfo เป็นไดเร็คทฟที่ควบคุมการโพรเซสไฟล์ Indexes ไดเร็คทีฟต่าง ๆ ที่เกี่ยวข้องกับการแสดงรายการของไฟล์

	Limit	เช่น IndexOption AddDescription, DirectoryIndex เป็นต้น ใช้เพิ่มความปลอดภัยในส่วนของไคลเอนท์ที่จะเข้ามาใช้งาน โดยสามารถกำหนดเป็นโฮสต์หรือ IP ได้ ซึ่งจะใช้ร่วมกับไดเร็คทีฟ Order, Allow, Deny ดังตัวอย่าง <Limit> Order deny, allow Deny from one.other.com</Limit>
	Options	ใช้สำหรับไดเร็คทีฟที่สนับสนุน miscellaneous options เช่น ContentDigest, XbitHack เป็นต้น
	All	กำหนดให้ใช้ AllowOverride ทั้ง 5 ประเภทข้างต้น
	None	กำหนดให้ AllowOverride ไม่มีการใช้งาน
Order	ระบุลำดับก่อนหลังในการกำหนดสิทธิ์ระหว่าง deny กับ allow	

ตารางที่ 3 : แสดง DIRECTIVE ในการกำหนดเกี่ยวกับไดเร็คทอรี

สำหรับตัวอย่างไดเร็คทอรีที่ต้องการใช้ เราสามารถกำหนดได้ดังนี้

```
<Directory "/www/docs-web/asia">
        AllowOverride AuthConfig
        Order allow,deny
        Allow from all
</Directory>
```

จากตัวอย่างเป็นการกำหนดให้มีการตรวจสอบว่าเป็นผู้มีสิทธิเข้ามาใช้หรือเปล่า แต่ไม่จำกัดว่าจะต้องเข้ามาใช้จากเครื่องไหน

AllowOverride AuthConfig : เป็นการกำหนดให้มีการตรวจสอบผู้เข้ามาใช้งาน โดยอ่านไฟล์ "htaccess"

Order allow, deny : เป็นลำดับของกฎในการกำหนดสิทธิ์การเข้ามาใช้ไดเร็คทอรี

Allow from all : เป็นกฎของ Allow ยอมให้ทุกเครื่อง (all) เข้ามาใช้ไดเร็คทอรีนี้ได้

รู้จักกับโมดูล **mod_alias**

Directive	คำอธิบาย
Alias	การสร้างชื่อปลอมในการเข้าถึงไดเร็คทอรีจริงบนเซิร์ฟเวอร์ ไม่ว่าจะเข้าไปใช้ไฟล์หรือเรียกเว็บเพจ
ScripAlias	การสร้างชื่อปลอมในการเข้าถึงไดเร็คทอรีจริงบนเซิร์ฟเวอร์เหมือนกับ Alias ต่างกันตรงที่ไฟล์ที่อยู่ในไดเร็คทอรีจริงจะเป็นสคริปต์ไฟล์ หรือไฟล์ที่เซิร์ฟเวอร์จะเรียกขึ้นมา เมื่อมีการเรียกใช้จากไคลเอนท์ เช่น /cgi-bin/

การสร้าง Alias

ในบางครั้งเรามีข้อมูลเก็ฐไว้ภายใต้ DocumentRoot แต่อยู่ในไดเร็คทอรีย่อยหลายชั้น ซึ่งการจะเข้าถึงข้อมูลโดยใส่ไดเร็คทอรีย่อยมาก ๆ คงไม่สะดวก หรือในกรณีที่เราต้องการใช้ข้อมูลที่อยู่นอก DocumentRoot ไคลเอนท์จะไม่สามารถเข้าถึงได้เลย เพราะอยู่นอกไดเร็คทอรีที่เซิร์ฟเวอร์ยอมให้ไคลเอนท์เข้าถึงได้

ในการแก้ปัญหาตรงนี้ สำหรับอาปาเช่มีการทำ alias เพื่อเข้าถึงไดเร็คทอรีที่ต้องการได้ ตัวอย่างเช่น ถ้าเราต้องการให้มีการเข้าถึงเว็บที่อยู่ภายใต้ไดเร็คทอรี "/www/web2/asia" แต่ DocumentRoot เป็น "/www/docs-web/" เราสามารถสร้าง alias ให้ชี้ไปถึงไดเร็คทอรีนี้ ในที่นี้สมมติชื่อเป็น "asia-zone" จะได้

```
Alias /asia-zone/ "/www/web2/asia/"          (กำหนด alias ชื่อ /asia-zone/)
    <Directory "/www/web2/asia">
        Options Indexes FollowSymlinks MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

การสร้าง Script Alias

ในบางครั้งเว็บเพจของเรามีการเรียกใช้สคริปต์ไฟล์ เช่น CGI. Prel เป็นต้น เราสามารถกำหนดไดเร็คทอรีที่เก็บสคริปต์เหล่านั้น โดยกำหนด script alias ชี้ไปยังไดเร็คทอรีที่ต้องการ ดังตัวอย่าง

```
ScriptAlias /cgi-bin "/www/cgi-bin/"  (กำหนด script alias ชื่อ /cgi-bin/ ชี้ไปที่ /www/cgi-bin)
    <Directory "/www/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>
```

การกำหนดชื่อ Alias หรือ script alias จากตัวอย่างเรากำหนด alias เป็น "/asia-zone/" มีเครื่องหมาย / ต่อท้าย ดังนั้นเมื่อไคลเอนท์เข้ามาใช้ผ่านบราวเซอร์จะต้องระบุ "http://localhost/asia-zone/" ให้มีเครื่องหมาย / ต่อท้ายเช่นกัน

และถ้าอีกจุดหนึ่งคือ ถ้าชื่อ alias หรือ script alias มีเครื่องหมาย "/" ต่อท้าย ไดเร็คทอรีจริงที่เราให้ alias ชี้มาจะต้องมีเครื่องหมาย "/" ต่อท้ายด้วยเช่นกัน นั่นคือถ้าไม่มีก็ต้องไม่มีทั้งคู่ด้วย

รู้จักกับโมดูล mod_autoindex

Directive	คำอธิบาย
IndexOptions	กำหนดว่าจะใช้ Fancy index หรือ standard เช่น IndexOptions FancyIndexing
AddIcon	กำหนดว่าไอคอนไหนใช้แสดงแทนไฟล์ชนิดไหน เช่น AddIcon /icons/binary.gif .bin .exe
DefaultIcon	กำหนดไอคอนสำหรับชนิดของไฟล์ที่ไม่ได้กำหนดไว้ก่อนด้วย AddIcon เช่น DefaultIcon/icons/rnknown.gif
ReadmeName	กำหนดชื่อไฟล์ Readme ให้เซิร์ฟเวอร์รู้จัก เช่น ReadmeName README
HeaderName	กำหนดชื่อไฟล์ Header ให้กับไดเร็คทอรี indexes เช่น HeaderName HEADER
IndexIgnore	กำหนดรายชื่อไฟล์ที่เซิร์ฟเวอร์ไม่ต้องสนใจ และไม่นำมาแสดงให้เห็นด้วย เช่น IndexLgonore .??* *~ *# HEADER* README* RCS CVS *,v *,t

รู้จักกับโมดูล mod_mime

Directive	คำอธิบาย
AddCoding	กำหนดให้บราวเซอร์ที่สนับสนุน สามารถขยายนยไฟล์ที่อยู่ในรูปบีบอีดได้เลย คล้ายกับการกำหนดโปรแกรมในการขยายไฟล์บีบอัดให้บราวเซอร์รู้ เช่น AddEncoding x-compress Z addEncoding gzip gz tgz
AddLanguage	เพิ่มภาษาสนับสนุนกับเอกสาร เช่น AddLanguage en .en
LanguagePriority	กำหนดลำดับก่อนหลังในการแสดงภาษา เช่น LanguagePriority en da nl et fr de el it ja kr no pl pt
AddType	การเพิ่มชนิดของ mime โดยไม่ต้องแก้ไขไฟล์ mime.types เช่น AddType application/x-tar tgz
AddHandler	กำหนด handler ให้กับชนิดของไฟล์ต่าง ๆ เช่น AddHandler cgi-script .cgi

ส่วนที่ 4 : เวอร์ชวลโฮสต์ (Virtual Host)

เวอร์ชวลโอสต์เป็นการติดตั้งเซิร์ฟเวอร์มากกว่าหนึ่งเซิร์ฟเวอร์บนเครื่องเดียวกัน นั่นคือเราสามารถทำเว็บสำหรับหลาย ๆ บริษัทให้อยู่บนเครื่องเดียวกันได้ ซึ่งก็เหมือนกับเว็บโฮสต์ติ้งนั่นเอง ที่ให้เช่าเนื้อที่บนเครื่องเซิร์ฟเวอร์ โดยผู้ที่มาเช่าสามารถเปิดให้บริการเว็บสำหรับองค์กรหรือเว็บเซิร์ฟเวอร์ของตนเองได้สำหรับในรายละเอียดของเวอร์ชวลโฮสต์จะพูดถึงในหัวข้อถัดไป

การติดตั้งระบบการขนส่งไฟล์ผ่านเน็ตเวิร์ค ZFTP

FTP หรือ File Transfer Protocol เป็นโปรโตคอลที่เราใช้ในการขนส่งไฟล์ระหว่างเครื่อง 2 เครื่อง ซึ่งโปรโตคอล FTP นั้นก็ทำงานอยู่บน TCP/IP อีกทีหนึ่ง เพราะฉะนั้นระบบที่ใช้ TC/IP กันอยู่แล้วจึงสามารถติดต่อกันได้ ไม่ว่าระบบปฏิบัติการจะแตกต่างกันก็ตาม

ตัวอย่างที่เราเห็นกันบ่อย ๆ คือ เว็บไซต์ที่ขึ้นต้นด้วย ftp:// ก้คือการเข้าไปใช้บริการ FTP Server ทำให้การแชร์ไฟล์ต่าง ๆ ให้เราเข้าไปดาวโหลดกัน ซึ่งไม่จำกัดว่าต้องเป็นระบบปฏิบัติการเดียวกันกับเซิร์ฟเวอร์ ขอเพียงใช้ TCP/IP โปรโตคอลก็ใช้ได้แล้ว

ลักษณะการทำงานของ **FTP**

จะแบ่งออกเป็นส่วนให้บริการ (FTP Server) กับส่วนที่ใช้บริการ (FTP Client) โดย FTP Server นั้นจะกำหนดทรัยพยากรที่ต้องการให้กับผู้ใช้บริการทั่วไปเข้ามาดาวน์โหลด หรือผู้ใช้บริการสามารถอัพโหลดข้อมูลมาเก็บไว้ก็ได้ โดยสามารถกำหนดสิทธิตามผู้ใช้ได้ ซึ่งจะต่างกับ MFS ตรงที่เครื่องไคลเอนท์ที่ใช้บริการ NFS จะไม่ได้ดาวน์โหลดทรัพยากรไป แต่จะเอาทรัพยากรบนเครื่องเซิร์ฟเวอร์มาไว้ที่เครื่องเสมือนเป็นทรัพยากรบนเครื่อง

เริ่มติดตั้ง **FTP Server**

โปรแกรม FTP โดยทั่วไปเราจะติดตั้งพร้อมกับตอนติดตั้งระบบปฏิบัติการแล้ว แต่อาจไม่ใช่เวอร์ชันใหม่นัก ซึ่งเราเองก็สามารถหาดาวน์โหลดเวอร์ชันใหม่มาใช้งานได้

ถ้าเป็นโปรแกรม FTP ของ wu-ftp เราสามารถเข้าไปค้นหาขอมูลเพิ่มเติมพร้อมทั้งดาวน์โหลดเวอร์ชันใหม่ได้ที่เว็บไซต์ www.wu-ftpd.org แต่ถ้าในตอนติดตั้งระบบปฏิบัติการีนกซ์เราไม่ได้ติดตั้ง FTP Server เราสามารถติดตั้งทีหลังได้จากซีดีลีนุกซ์แผ่นที่ 1 ภายใต้ไดเร็คทอรี redhat/RPMS/ จะพบว่ามีแพ็คเกจของ wu-ftp เช่น wu-ftp-2.6.0-i386.rpm เป็นต้น

การติดตั้งโปรแกรม FTP แบบแพ็คเกจด้วยโปรแกรม rpm

ในที่นี้เราจะพูดกันถึงการติดตั้งทั้งแบบแพ็คเกจ และแบบที่เป็นโค้ดต้นฉบับกัน ดังนี้

1. เรียกคำสั่ง rpm ดังต่อไปนี้

[root @ redhat RPMS] # rpm -vi wu-ftp-2.6.0-i386.rpm

2. เมื่อติดตั้งไปแล้วถ้าอยากรู้ว่ามีการติดตั้งไฟล์อะไรบ้าง ให้เรียกคำสั่งต่อไปนี้

[root @ redhat RPMS] # rpm -ql wu-ftp-2.6.0-i386.rpm

3. การติดตั้งแบบแพ็คเกจเมื่อติดตั้งเสร็จแล้ว ก็พร้อมใช้งาน FTP Server ได้เลย เพราะทุกอย่างถูกเตรียมไว้หมดแล้ว เหลือเพียงแต่กำหนดไดเร็คทอรีเพื่อให้บริการเท่านั้นเอง

ติดตั้งโปรแกรม FTP แบบโค้ดต้นฉบับ

มีขั้นตอนในการติดตั้งดังนี้

1. เริ่มด้วยการแตกไฟล์ tar ก่อน ซึ่งจะได้ไดเร็คทอรีของ wu-ftpd-version
2. จากนั้นให้เปลี่ยนไดเร็คทอรี แล้วเข้าไปคอมไพล์ดค้ดต้นฉบับด้วย build Script

 # ./build < รหัสของระบบปฏิบัติการ >

ในที่นี้เราคอมไพล์ระบบปฏิบัติการลีนุกซ์ ดังนั้นจึงเลือกใช้รหัส Inx ซึ่งเราจะเรียก build script ดังนี้ แล้วรอจนกว่าจะคอมไพล์เสร็จ

 # ./build Inx

3. ต้องการติดตั้งแบบอัพเกรดหรือว่าติดตั้งใหม่หมด ในกรณีที่ต้องการติดตั้งแบบอัพเกรด เราต้องสำรองไฟล์ configuration ต่าง ๆ ที่เกี่ยวข้องกับ FTP โดยทั่วไปจะอยู่ใต้ไดเร็คทอรี /etc/ และถ้าต้องการเก็บเวอร์ชันเก่าไว้ด้วย ก็ให้สำเนาโปรแกรม FTP เวอร์ชันเก่าด้วย โดยส่วนใหญ่จะเก็บอยู่ภายใต้ไดเร็คทอรี /usr/sbin/
4. ติดตั้งตัวโปรแกรมด้วย build script

 # ./build install

ตรวจสอบพอร์ตบริการ

เมื่อติดตั้งเสร็จแล้ว ต้องตรวจสอบว่ามีการกำหนดพอร์ตสำหรับการบริการของ FTP หรือยัง โดยตรวจสอบได้ที่ไฟล์ /etc/services ว่ามีบรรทัดดังต่อไปนี้หรือเปล่า ถ้าไม่มีให้ทำการเพิ่มเข้าไปด้วย

ftp 21/tcp

ftp 21/udp

กำหนดการให้บริการของ FTP

เมื่อกำหนดพอร์ตเรียบร้อยแล้ว เราก็มากำหนดการให้บริการกัน โดยเราต้องทำการสร้างไฟล์บริการของ FTP ขึ้นมาก่อนที่ไดเร็คทอรี่ /etc/xinetd.d/ สมมติว่าตั้งชื่อเป็น wu-ftpd ซึ่งภายในไฟล์จะมีการกำหนดดังต่อไปนี้

```
#default: on
#description: The wu-ftpd FTP server serves FTP connections. It uses \
#normal, enencrypted usernames and passwords for authentication.
Service ftp
{
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l –a
        log_on_failure          += DURATION USERID
        nice                    = 10
        disable                 = no
}
```

ในกรณีที่ติดตั้งโปรแกรม FTP แบบแพ็คเกจ จะพบว่ามีการติดตั้งไฟล์นี้มาให้แล้ว

เปิดให้บริการ FTP

เมื่อทำครบ 3 ขั้นตอนข้างต้นแล้ว เราก็สามารถเปิดให้บริการ FTP ได้แล้วโดยเรียกคำสั่งดังต่อไปนี้

```
[ root @ redhat root ] # /init.d/xinetd restart
Stopping xinetd : [ OK ]
Starting xinetd : [ OK ]
```

หลังจากเลือกสคริปแล้ว คงต้องตรวจสอบดูก่อนว่าพอต FTP นั้นเปิดให้บริการหรือยังโดยเรียกคำดังนี้

```
[ root @ redhat root] # netstart -na i grep 21
tcp       0      0 0.0.0.0:21              0.0.0.0:*LISTEN
```

ถ้ามีบรรทัดตัวอย่างให้เห็น แสดงว่าพอร์ตได้กำหนดพร้อมให้บริการแล้ว คราวนี้ก็มาดูว่าในการล็อกอินเพื่อเข้ามาในบริการนั้น สามารถทำได้กี่แบบ และแต่ละแบบเป็นแบบไหน

ประเภทของล็อกอินในบริการ FTP

ในการเข้ามาใช้บริการ FTP บนเครื่อง FTP Server เราจำเป็นต้องมีการตรวจสอบสิทธิ์ผู้เข้ามาใช้บริการด้วยการล็อกอิน โดย FTP Server แบ่งชนิดของการล็อกอินเข้ามาเป็น 3 แบบด้วยกัน ดังนี้

1. ล็อกอินด้วยผู้ใช้ที่มีอยู่ในระบบ (Real FTP) : การล็อกอินแบบนี้ ผู้ที่ล็อกอินเข้ามาใช้บริการจะเป็นผู้ใช้ที่มีอยู่จริงบนเครื่องเซิร์ฟเวอร์ด้วย นั่นคือมีบัญชีผู้ใช้อยู่ที่ไฟล์ /etc/passwd มีรหัสผ่านเป็นของตนเอง ดังนั้นเมื่อผู้ใช้ล็อกอินด้วยบัญชีผู้ใช้บนระบบ เขาจะมีสิทธิ์เทียบเท่ากับสิทธิ์ที่เซิร์ฟเวอร์ก็ได้ภายในเซิร์ฟเวอร์รายนั้น ๆ ซึ่งเมื่อล็อกอิเข้ามาแล้ว สามารถจะเปลี่ยนไดเร็คทอรีไปที่ไหนก็ได้ภายในเซิร์ฟเวอร์ นั่นถือเป็นความเสี่ยงอย่างหนึ่งด้วย
2. ล็อกอินด้วยผู้ใช้ที่มีอยู่ในระบบแต่จำกัดขอบเขต (Guest FTP) : การล็อกอินแบบนี้คล้ายกับแบบที่ 1 นั่นคือจะล็อกอินเข้ามาด้วยบัญชีผู้ใช้ที่มีอยู่ในเซิร์ฟเวอร์ พร้อมทั้งรหัสของบัญชีผู้ใช้นั้น ๆ แต่จะแตกต่างกันตรงที่เมื่อผู้ใช้บริการล็อกอินเข้ามาแล้ว จะไม่สามารถเปลี่ยนไดเร็คทอรีไปไหนได้เกินขอบเขตที่เซิร์ฟเวอร์กำหนด เพราะเซิร์ฟเวอร์ได้จำลองไดเร็คทอรีรูท * / * เป็นที่อื่นแทน จึงสามารถกำหนดขอบเขตของการเปลี่ยนไดเร็คทอรีได้ ไม่ให้เข้าไปยุ่งกับไดเร็คทอรีอื่นในระบบได้ จึงถือว่าเป็นการเพิ่มความปลอดภัยให้กับเซิร์ฟเวอร์ด้วย
3. ล็อกอินโดยผู้ใช้ที่ไม่อยู่ในระบบ (Anonymos FTP) : การล็อกอินแบบสุดท้ายนี้เป็นแบบที่มีใช้เยอะอยู่เหมือนกัน ยิ่งถ้าเราติดตั้ง FTP Server ให้บริการกับผู้ใช้ทั่วไป เช่น เปิดเสรีให้ผู้ใช้ทั่วโลกเข้ามาใช้บริการ การจะมาสร้างบัญชีผู้ใช้ให้รองรับกับคนทั่วโลกคงไม่มีใครทำกันแน่ การล็อกอินในแบบนี้จึงทำขึ้นมาเพื่อให้ผู้ใช้บริการที่ไม่มีบัญชีผู้ใช้อยู่บนเซิร์ฟเวอร์เข้ามาใช้บริการได้ โดยล็อกอินเข้ามาด้วย anonymous หรือ ftp และระบุรหัสผ่านเป็น e-mail address ของผู้เข้ามาใช้บริการเท่านี้ก็สามารถใช้บริการได้แล้ว ซึ่งในแบบนี้ยังมีบางส่วนคล้ายกับแบบที่ 2 คือ จะมีการกำหนดขอบเขตของไดเร็คทอรีที่สามารถเข้าไปได้

การติดตั้งล็อกอินในบริการ FTP

สำหรับวิธีการติดตั้งล็อกอินในบริการ FTP นั้นแยกอธิบายตามประเภทของการล็อกอินได้ดังนี้

ล็อกอินด้วยผู้ใช้ที่มีอยู่ในระบบ (Real FTP)

การล็อกอินประเภทนี้ก็ทำเหมือนกับการสร้างบัญชีผู้ใช้ทั่วไปบนระบบปฏิบัติการ ดังขั้นตอนต่อไปนี้

1. สร้างบัญชีผู้ใช้ที่ไฟล์ /etc/passwd

Sommut:*:234:100:Sommut Namfang:/home/sommut:/bin/ksh

2. กำหนดรหัสผ่านให้กับผู้ใช้ใหม่

[root @ redhat root] # pwconv

[root @ redhat root] #passwd sommut

3. สร้างโฮมให้กับผู้ใช้

[root @ redhat root] # mkdir /home/sommut

[root @ redhat root] # chown 234:100 /home/sommut

เพียงแค่นี้เราก็มีบัญชีผู้ใช้รายใหม่เพิ่มแล้ว ซึ่งสามารถล็อกอินเข้ามาใช้บริการได้แล้ว

ล็อกอินด้วยผู้ใช้ที่มีอยู่ในระบบแต่จำกัดขอบเขต (Guest FTP)

การสร้างล็อกอินในแบบนี้จะแตกต่างจากแบบแรกตรงที่เราจะกำหนดขอบเจตขอลไดเร็คทอรีไว้โดยการจำลองไดเร็คทอรีรูทใหม่ เช่น กำหนดให้โอมของผู้ใช้รายนั้นเป็นรูท ก็จะทำให้ผู้ที่ล็อกอินเข้ามาแบบ Guest FTP จเปลี่ยนไดเร็คทอรีออกไปจากโฮมไม่ได้ ดังขั้นตอนต่อไปนี้

1. สร้างบัญชีผู้ใช้ที่ไฟล์ /etc/passwd

Ftpacct:*:235:200: FTP Account :/home/ftpacct/./:/etc/ftponly

จากตัวอย่างที่ไฟล์ " /./ " เพิ่มเข้ามานั่นคือการใช้ฟังก์ชัน chroot() ในการจำลองไดเร็คทอรีรูทใหม่ โดยจากตัวอย่างเป็นการจำลองไดเร็คทอรีรูทให้อยู่ภายใต้ไดเร็คทอรีโฮมคือ /home/ftpacct/

ถ้าเราต้องการจำลองไดเร็คทอรีรูทให้อยู่ภายใต้ /home/ และสามารถเปลี่ยนไดเร็คทอรีไปยังไดเร้คทอรีย่อย ./ftpacct/ ให้เราแก้ไฟล์ /etc/passwd เป็นดังนี้

ftpacct*:235:200: FTP Account :/home/./ ftpacct/:/ etc/ftponly

2. กำหนดรหัสผ่านให้กับผู้ใช้ใหม่

 # pwconv

 # passwd ftpacct

3. สร้างกลุ่มของผู้ที่ไฟล์ /etc/group ซึ่งเมื่อเราสร้างบัญชีผู้ใช้ขึ้นมา ต้องตรวจสอบว่ากลุ่มผู้ใช้เป็นกลุ่มใหม่หรือเปล่า ถ้าใช่ก็ต้องเข้ามาสร้างกลุ่มผู้ใช้ด้วยดังนี้

client::200:fpacct

4. สร้างโฮมไดเรทอรีสำหรับผู้ใช้ใหม่

 # mkdir /home/ftpacct

 # chown ftpacct:client /home/ftpacct (กำหนดเจ้าของไฟล์ และสิทธิ์ในการเข้าถึงไฟล์

 # chmod 755 /home/ftpacct

5. กำหนดไดเร็คทอรีภาใต้ไดเร็คทอรีจำลอง

 # cd /home/ftpacct

 # mkdir etc lib bin

 # chow root:root /home/ftpacct

 # chmod 111 /home/ftpacct/etc

chmod 111 /home/ftpacct/bin

 # chow root:root bin/*

#chmod 111 bin/*

6. สำเนาไฟล์ที่จำเป็นมาไว้ยังไดเรคทอรี่ย่อย bin lib

 # cd /home/ftpacct

 # cp /bin/ls bin

 # cp /bin/gzip bin

 # cp /bin/tar bin

#chown root:root bin/*

#chmod 111 bin/*

สำหรับการสำเนาไฟล์ที่เป็นไลบรารีไฟล์ จำเป็นต้องตรวจสอบดูจากไฟล์ไบนารี่ ที่เราต้องการนำมาใช้ภายใต้ไดเร็คทอรี bin โดยใช้คำสั่ง ldd ในการตรวจสอบดังนี้

[root @ redhat root] # ldd /bin/ls

libtermcap.so.2 => /lib/libtermcap.so.2 (0x40021000)

libc.so.6 => /lib/libc.so.6 (0x40025000)

/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)

จากตัวอย่างพบว่าคำสั่ง ls มีการอ้างอิงถึงไลบรารีไฟล์ 3 ไฟล์ด้วยกัน ซึ่งเราพิจารณาจากไฟล์ทางซ้ายมือว่าเป็นไฟล์ชื่ออะไรบ้าง เครื่องหมาย => นั่นหมายถึงเป็นลิงค์ไฟล์ โดยทั่วไปถ้าเป็นไลบรารีของระบบปฏิบัติการจะอยู่ภายใต้ไดเร็คทอรี /lib/ ดังนั้นให้เราเปลี่ยนไดเร็คทอรีไปที่ /lib/ แล้วเรียกคำสั่ง

ls -l < ชื่อไบรารีไฟล์ >

พบว่าไฟล์ที่เจอเป็นลิงค์ไฟล์ ดังนั้นให้เราทำสำเนาไฟล์ที่เป็นต้นฉบับของลิงค์ไฟล์ ไปไว้ยังไดเร็คทอรี /home/ftpacct/lib/ แล้วทำให้เหมือนกันดังนี้

สำเนาไฟล์ต้นฉบับไปยังไดเร็คทอรี lib จำลอง

 # cp /lib/libtermcap.so.2.0.8

 # cp /lib/libc-2.2.4.so /home/ftpacct/lib

 # cp /lib/ld-2.2.4.so /home/ftpacct/lib

สร้างลิงค์ไฟล์ให้เหมือนกับต้นฉบับ

 # cd /home/ftpacct/lib

 # in -s libtermcap.so.2.0.8 libtermcap.so.2

 # in -s libc-2.2.4.so libc.so.6

 # in -s ld-2.2.4.so lb-linux.so.2

กำหนดย่อของไฟล์และสิทธิ์การใช้ไฟล์ให้ตรงกับต้นฉบับ

 # chown root:root libtermcap.so.2.0.8 lidc-2.2.4.so lb-2.2.4.so

 # chmod 755 libtermcap.so.2.0.8 libc-2.2.4.so ld-2.2.4.so

จากตัวอย่างข้างต้นเป็นการตรวจสอบเฉพาะคำสั่ง ls แต่ในตัวอย่างข้างต้นได้มีการสำเนาไฟล์ /bin/gzip และ /bin/tar มาด้วย ดังนั้นเราต้องตรวจสอบไลบรารีสำหรับไฟล์ทั้ง 2 ดังนี้

[root @ redhat root] # ldd /bin/gzip

libc.so.6 => /lib/libc.so.6 (0x40025000)

/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)

[root @ redhat root] # ldd /bin/tar

libc.so.6 => /lib/libc.so.6 (0x40025000)

/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)

พบว่ามีการใช้ไลบรารีเหมือนกับคำสั่ง ls ดังนั้นเราก็ไม่ต้องทำสำเนาเพิ่มอีก

การสร้างไดเร็คทอรีย่อย etc, bin, lib จะต้องสร้างขึ้นภายใต้ไดเร็คทอรีรูทจำลองเช่น ถ้าเรากำหนดในไฟล์ /etc/passwd เป็น

Ftpacct:*:235:200:FTP Account:/home/:/etc/ftponly

นั่นคือไดเร็คทอรีรูทจำลองจะเป็น /home/ ดังนั้นไดเร็คทอรีย่อย etc, bin, lib จะต้องสร้างไว้ภายใต้ไดเร็คทอรี /home/

7. สร้างไฟล์ passwd และ group จำลอง สำหรับไดเร็คทอรีรูทจำลอง ดังนี้

 # cd /home/ftpacct/etc

สำหรับไฟล์ /home/ftpacct/etc/passwd

 # vi passwd

root:*:0:0:root:/:/etc/ftponly (โดยเพิ่มบรรทัดดังต่อไปนี้)

ftpacct:*:235:400: FTP Account :/home/ftpacct/./:/etc/ftponly (โดยเพิ่มบรรทัดดังต่อไปนี้)

สำหรับไฟล์ /home/ftpacct/etc/group

 # vi group

Root::0:root

Client::200:ftpacct

กำหนดเจ้าของไฟล์ และสิทธิ์ในการเข้าถึงไฟล์

 # chown root:root passwd group

 # chmod 444 passwd group

8. แก้ไขไฟล์ /etc/ftpaccess ซึ่งถ้าติดตั้งโปรแกรม FTP Server แบบแพ็คเกจ ไฟล์นี้จะถูกสำเนามาไว้ที่ /etc/ ให้โดยอัติโนมัติ แต่ถ้าเราติดตั้งโปรแกรม FTP Server จากโค้ดต้นฉบับ เราสามารถสำเนาอย่างไฟล์ configuration ต่าง ๆ ที่เกี่ยวข้องกับการติดตั้ง FTP Server ได้ที่ไดเร็คทอรีทที่เราแตก tar file ออกมา แล้วเปลี่ยนไดเร็คทอรีเข้าไปที่ ./doc/example/ เมื่อได้ไฟล์ ftpaccess แล้วแก้ไขดังตัวอย่างต่อไปนี้

กำหนด User class ให้เพิ่ม guest g-hkwx

Class all real,anonymous,guest

เพิ่มความปลอดภัยเกี่ยวกับการใช้คำสั่งต่าง ๆ นี้ โดยระบุ guest เข้าไปด้วย

delete no anonymous,guest

overwrite no anonymous,guest

rename no anonymous,guest

chmod no anonymous,guest

umask no anonymous,guest

กำหนด path-filter ให้กับ guest-ftp (จะกำหนดหรือไม่ก็ได้)

path-filter guest /etc/pathmsg ^[-A-Za-z0-9_\.]*$ ^\. ^_

กำหนด guest group โดยเพิ่มกลุ่มของผู้ใช้แบบ guest-ftp เข้าไปด้วย จากตัวอย่างเพิ่มกลุ่มผู้ใช้ client guestgroup ftpchroot client

หลังจากเสร็จขั้นตอน เราก็สามารถเข้ามาใช้งานแบบ guest FTP ได้แล้ว

ล็อกอินด้วยผู้ใช้ที่ไม่มีอยู่ในระบบ **(Anonymous FTP)**

สำหรับการสร้างล็อกอินแบบสุดท้ายนี้ จะคล้ายกับการสร้างล็อกอินแบบ guest FTP นั่นคือมีการจำลองไดเร็คทอรีรูท และจำกัดขอบเขตของการเข้าถึงข้อมูลภายในเครื่องให้อยู่ภายใต้ไดเร็คทอรีที่กำหนดส่วนข้อต่างคือ แบบ Guest FTP 1 ล็อกอินต้องกำหนด 1 รหัสผ่าน แต่แบบ anonymous FTP มาพร้อมติดตั้งได้ 2 แบบคือ

ติดตั้งแบบแพ็คเกจ : โดยปกติตอนติดตั้งระบบปฏิบัติการ ถ้ามีการิดตั้ง FTP Server ก็จะติดตั้ง anonftp-version.rpm หรือสามารถดาวน์โหลดจากอินเตอร์เน็ตก็ได้ การติดตั้งก็เรียกโปรแกรม rpm โดยใช้คำสั่งดังนี้

[root @ redhat root]# rpm -q anonftp

ถ้าไม่สามารถติดตั้งได้จากแผ่นลีนุกซ์แผ่นที่ 1 โดยเข้าไปที่ไดเร็คทอรี Redhat/RPMs/ จะพบแพ็คเกจ anonftp-version.rpm หรือสามารถดาวน์โหลดจากอินเตอร์เน็ตก็ได้ การติดตั้งก็เรียกโปรแกรม rpm โดยใช้คำสั่งดังนี้

[root @ redhat root]# rpm -iv anonftp-4.0-9.rpm

แพ็คเกจจะติดตั้งไฟล์ต่าง ๆ ทีไดเร้คทอรี /var/ftp/ ซึ่งถือว่าเป็นไดเร็คทอรีรูทจำลองของ

Anonymous user โดยมีไฟล์ต่าง ๆ และทำการอัพเดตไฟล์ /etc/passwd บรรทัดบัญชีผู้ใช้ "ftp" ดังตัวอย่างนี้

ftp:*:14:50: FTP User :/var/ftp/:/sbin/nologin

แต่ถ้าเราต้องการลงมือติดตั้งเอง ก็ลองดูในหัวข้อถัดไปครับ

ติดตั้งโดยแก้ไขไฟล์ด้วยตัวเอง : ในกรณีที่ต้องการติดตั้ง anonymous FTP เอง ก็ทำได้ไม่ยาก ดังขั้นตอนต่อไปนี้

1. ตรวจสอบไฟล์ /etc/passwd : ให้ตรวจสอบดูว่ามีบัญชีผู้ใช้ชื่อ ftp หรือยัง ถ้ายังให้สร้างขึ้นมาด้วย แต่โดยปกติจะมีการสร้างเตรียมไว้ให้อยู่แล้ว ดังเช่น

   ftp:*:14:50: FTP User :/home/ftp/: /sbin/nologin

2. ตรวจสอบไฟล์ /etc/group : โดยรวจสอบว่ามีการสร้างกลุ่มผู้ใช้ ftp หรือยัง ถ้ายังให้สร้างขึ้นมาด้วย โดยปกติกลุ่มผู้ใช้ก็จะถูกสร้างเตรียมไว้อยู่แล้ว ดังเช่น

   ftp:50: ftp

3. เตรียมไดเร็คทอรีรูทจำลอง : ตัวอย่างเรากำหนดให้ไดเร็คทอรีรูทเป็น /home/ftp/ ดังนั้นให้เราสร้างไดเร็คทอรีนี้ขึ้นมา พร้อมทั้งกำหนดเจ้าของและสิทธิ์

   # mkdir /home/ftp

   # chown root:root /home/ftp

   # chmod 755 /home/ftp

4. สร้างไดเร็คทอรีย่อยภายใต้ไดเร็คทอรีรูทจำลอง : ขั้นตอนนี้ก็เหมือนกับการทำ Guest FTP นั่นคือสร้างไดเร็คทอรี etc, bin, lib และทำการคัดลอกไฟล์เฉพาะที่จำเป็นมาใช้ โดยสร้างไดเร็คทอรีย่อยก่อน ดังนี้

   # cd /home/ftp

   # mkdir etc bin lib

   # chown root:root etc bin lib

   # chmod 111 etc bin

   # chmod 755 lib

จำลองไฟล์ passwd, group ภายใต้ /home/ftp/etc

 # vi passwd

root:*:0:0:root:/: **(โดยเพิ่มบรรทัดดังต่อไปนี้)**

ftp:*:14:50: FTP Account :/home/ftp:

 # vi group

root::0:root **(โดยเพิ่มบรรทัดดังต่อไปนี้)**

ftp:50:ftp

 # chown root:root passwd group

 # chmod 444 passwd group

สำเนาคำสั่งที่จำเป็นไว้ที่ /home/ftp/bin

 # cp /bin/is

 # cp /bin/gzip

 # cp /bin/tar

 # chown root:root ./*

 # chmod 111 ./*

สำเนาไลบรารีไฟล์ที่จำเป็นและทำลิงค์ภายใต้ /home/ftp/lib ให้ใช้คำสั่ง ldd ตรวจสอบดูว่ามีไลบรารีไฟล์อะไรบ้าง

ซึ่งเมื่อดูแล้วขั้นตอนการทำก็คล้าย ๆ กับการติดตั้งแบบ Guest FTP เลย ก็คงไม่ยากเกินไปนะครับ เมื่อกำหนดการล็อกอินไว้บนเซิร์ฟเวอร์เสร็จแล้ว ก็มาดูกันว่าเราจะเข้ามาใช้บริการได้อย่างไร

การติดตั้งระบบการชนส่งจดหมายอิเล็กทรอนิคส์ (E-mail)

ในอดีตที่เทคโนโลยียังไม่พัฒนามากนัก การติดต่อที่สะดวกก็คงเป็นโทรศัพท์ ถ้าพูดกันถึงเรื่องการส่งเอกสารล่ะ ก็คงมีเครื่องโทรสาร แต่เครื่องโทรสารคงไม่มีใช้กันตามบ้านทั่วไป ดังนั้นระบบไปรษณีย์จึงเป็นหนทางหนึ่งที่เราใช้ติดต่อกัน ไม่ว่าจะเป็นโทรเลข พัสดุภัณฑ์ ธนาณัติ และที่ขาดไม่ได้คงเป็นจดหมาย ซึ่งโดยปกติจดหมายที่ส่งแบบธรรมดาภายในจังหวัดก็คงจะวันเดียว แต่ถ้าเป็นต่างจังหวัด และยิ่งเป็นอำเภอที่ไกลจากตัวเมืองก็ยิ่งต้องใช้เวลามากขึ้นไปด้วย ถ้าต้องการให้เร็วขึ้นก็มีบริการ EMS ซึ่งราคาก็จะแพงตามน้ำหนัก

ในปัจจุบันเมื่ออินเตอร์เตเข้ามามีบทบาทมากขึ้น การติดต่อกันจึงมีช่องทางมากขึ้นสะดวกขึ้น จากรูปของจดหมายที่เป็นกระดาษ ก็ผันเปลี่ยนมาอยู่ในรูปของจดหมายอิเล็กทรอนิกส์ หรือที่รู้จักกันคือ E-mail เป็นการติดต่อที่เป็นที่นิยมกันมากขึ้น ขอเพียงมีไฟฟ้า โทรศัพท์ และคอมพิวเตอร์ เราก็สามารถติดต่อกันไม่ว่ามุมไหนของโลกเพียงไม่กี่นาทีเท่านั้น ซึ่งในบทนี้จะกล่าวถึงลักษณะการทำงานของระบบรับส่งจดหมายอิเล็กทรอนิกส์และวิธีการติดตั้งเซิร์ฟเวอร์ที่ทำหน้าที่เหมือนที่ทำการไปรษณีย์กัน

ลักษณะการทำงานของระบบรับส่งอีเมล

จะว่าไปลักษณะการทำงานของระบบอีเมล ถ้าเทียบกับระบบไปรษณีย์แล้วก็คล้ายกันอยู่คือเริ่มต้นด้วยเราอยากเขียนจดหมายขึ้นมา เมื่อเขียนเสร็จก็จะส่งที่ตู้จดหมาย แล้วบุรุษไปรษณีย์ก็จะนำจดหมายไปรวมไว้ที่ที่ทำการไปรษณีย์ เพื่อส่งต่อไปยังที่ทำการไปรษณีย์ปลายทาง แล้วบุรุษไปรษณีย์ปลายทางก็จะนำไปส่งถึงผู้รับเป็นอันจบกระบวนการ

คราวนี้เมื่อเป็นอีเมล ผู้ส่งหรือ sender เริ่มเขียนจดหมาย เขียนเสร็จก็กดปุ่มส่งผ่านโปรโตคอลส่งไปยังเครื่องเมลเซิร์ฟเวอรืต้นทาง จากนั้นเมลเซิร์ฟเวอร์จะส่งไปยังเครื่องที่เป็นรีเลย์โฮสต์ต้นทาง

เนื่องจากรีเลย์โฮสต์เป็นเครื่องที่สามารถติดต่อกับโลกภายนอกได้ (โดยทั่วไปเครื่องเมล์เซิร์ฟเวอร์อาจทำหน้าที่เป็นรีเลย์โฮสต์ในเครื่องเดียวกันก็ได้) จากรีเลย์โฮสต์ต้นทางเมื่อได้รับเมลมาแล้ว จะติดต่อกับรีเลย์โฮสต์ปลายทางเพื่อส่งเมลบับนี้ไป เมื่อเมลไปถึงเมลเซร์ฟเวอร์ปลายทางเรียบร้อยแล้วถือเป็นอันจบกระบวนการส่งผล

คราวนี้เมื่อผู้ได้รับเช็คเมล ไมว่าจะใชวิธีไหนก็ตามก็จะต้องติดต่อกับเมลเซร์ฟเวอร์ของตนเองเพื่อนำเมลฉบับนั้นมาอ่าน ในที่นี้เครื่องเมลเซิร์ฟเวอร์ หรือเครื่องรีเลย์โฮสต์ก็จะทำหน้าที่เหมือนกับที่ทำการไปรษณีย์นั่นเอง

ศัพท์ที่ควรรู้เกี่ยวกับระบบอีเมล

เมื่อเข้าสู่ขั้นตอนการติดตั้ง เรามารู้จักกับศัพท์ที่ใช้กับระบบอีเมลกันสักนิด เพื่อความเข้าใจจ่ายขึ้นเมื่อมีการพูดถึงกันในหัวข้อถัดไป ดังนี้

Sender	ผู้ส่งจดหมาย
Recipient	ผู้รับปลายทาง
Mail Server	เครื่องที่มีกล่องรับเมลของผู้รับอยู่ โดยปกติจะเก็บไว้ที่ไดเร็คทอรี /var/mail/
Mail Client	เครื่องที่ทำการดึงเมลหรือ mount กล่องจดหมายจากเมลเซิฟเวอร์ เพื่ออ่านเมลในกล่องจดหมาย เช่น เครื่องที่ใช้ Outlook หรือ Netscape mail ในการดึงเมลมาอ่าน เป็นต้น
Relay Host	เครื่องที่ทำหน้าที่ติดต่อกับรีเลย์โฮสต์ปลายทาง เพื่อส่งจดหมายต่อไปให้เครื่องปลายทาง ถ้าเมลเซิร์ฟเวอร์ทำหน้าที่ติดต่อกับปลายทางเอง นั่นคือเมลเซิร์ฟเวอร์ก็ทำหน้าที่เป็นรีเลย์โฮสต์ไปด้วย
Mail Address	เปรียบเสมือนชื่อและที่อยู่ของผู้รับ ซึ่งแบ่งได้เป็น 3 แบบด้วยกัน Username ใช้ได้ในกรณีที่ผู้รับอยู่บนเมลเซิร์ฟเวอร์เดียวกันเท่านั้น Username@machine ใช้ได้ในกรณีผู้รับที่อยู่บนเมลเซิร์ฟเวอร์ภายในโดเมนเดียวกัน Username@machine.domain หรือ Username@domain ใช้ส่งหาผู้รับโดยระบุเป็นโดเมน ซึ่งสามารถส่งหาผู้รับต่างโดเมนกันได้ หรือจะเป็นดดเมนเดียวกันตามแต่จะระบุ
MTA	ย่อมาจาก Mail Transfer Agent เป็นโปรแกรมที่ใช้รับส่งเมลระหว่างเครื่อง 2 เครื่อง โปรแกรมส่วนใหญ่จะใช้ sendmail

SMTP คืออะไร

SMTP หรือ Simple Mail Transfer Protocol เป็นโปรโตคอลที่ใช้รับส่งอีเมลระหว่างเครื่องคอมพิวเตอร์ ไม่ว่าจะเป็นอินทราเน็ต หรืออินเตอร์เน็ตก็ตาม โดยที่ MTA จะใช้ SMTP ในการติดต่อระหว่างกัน

SMTP จะทำหน้าที่ในการกำหนดว่า MTAs แต่ละตัวจะติดต่อกันได้อย่างไรผ่านทาง TCP/IP จดหมายที่ส่งไปนั้นอาจจะส่งตรงไปยัง MTA ปลายทางเลย หรือว่าผ่าน MATs หลายเครื่อง (หมายถึงผ่านรีเลย์โฮสต์หลายเครื่อง) โดยผ่านกระบวนการ Store และ Forward ก็ได้เช่นกัน

โปรโตคอล SMTP จะไม่สนใจว่าข้อความจดหมายจะเป็นอะไร แต่จำกัดว่า SMTP สามารถส่งได้แต่ข้อมูลที่เป็นข้อคงวาม ASCII เท่านั้น ไม่สามรถส่งไฟล์ที่เป็นเพลง, หนัง, รูปภาพ หรืออื่น ๆ ได้ ซึ่งถ้าเราเราต้องการส่งไฟล์เหล่านี้ผ่านทาง SMTP จะต้องแปลงไฟล์เหล่านี้ให้อยู่ในรูปของข้อความก่อน และเมื่อส่งไปถึงปลายทางแล้ว ค่อยทำการแปลงกลับอีกที

สำหรับรายละเอียดเกี่ยวกับ SMTP เพิ่มเติมเราสามารถเข้าไปดูเอกสารของ RFC (Request For Comments) ซึ่งสามารถเข้าไปค้นหาได้ที่ http://www.rfc-editor.org โดย RFCs ที่เกี่ยวข้องกับ SMTP คือ RFC 821 และ RFC 1869

MIME ใช้ทำอะไร

เมื่อโปรโตคอล SMTP ยอมให้มีการส่งข้อมูลได้เฉพาะข้อความ ASCII ดังนั้นจึงมีการหาวิธีที่จะทำให้สามารถส่งไฟล์ชนิดอื่น ๆ ให้ได้ นั่นคือ MIME หรือ Multipurpose Internet Mail เป็นโปรแกรมที่คอยแปลงไฟล์ต่าง ๆ ให้อยู่ในรูปของข้อความ ASCII แล้วเมื่อถึงปลายทางโปรแกรม MIME ที่ปลายทางจะแปลงกลับให้เป็นไฟล์ชนิดเดิม

สำหรับรายละเอียดเกี่ยวกับ MIME เพิ่มเติมเราสามารถเข้าไปดูเอกสารของ RFC (Request For Comments) โดย RFCs ที่เกี่ยวข้องกับ SMTP คือ RFC 2046-2049

การติดตั้งโปรแกรม Sendmail บนเมลเซิร์ฟเวอร์

สำหรับโปรแกรมที่เป็น MTA นั้น นอกจาก Sendmail แล้วยังมี Qmail, smail ด้วยเป็นต้น แต่โดยทั่วไปเรามักจะใช้ Sendmail เป็น MTA ดังนั้นในที่นี้จะพูดถึงการติดตั้งโปรแกรม Sendmail แทนซึ่งโดยปกติเมื่อติดตั้งระบบปฏิบัติการลีนุกซ์แล้ว จะมีการติดตั้งโปรแกรม Sendmail มาด้วยอยู่แล้ว ดังนั้นก่อนติดตั้งโปรแกรมควรตรวจสอบก่อน ดังขั้นตอนต่อไปนี้

1. ตรวจหาโปรแกรม Sendmail : เราสามารถตรวจสอบได้ว่ามีโปรแกรม Sendmail ติดตั้งไว้แล้วหรือไม่ โดยใช้คำสั่ง rpm ว่ามีการลงโปรแกรม Sendmail แบบแพ็คเกจไว้หรือเปล่า โดยใช้คำสั่ง

rpm –q sendmail | sendmail |-8.11.6-3

ถ้าตรวจสอบด้วย rpm แล้วไม่พบแพ็คเกจของ Sendmail ให้ลองตรวจสอบดูว่ามีไฟล์ /usr/sbin/sendmail หรือว่า /usr/lib/sendmail หรือไม่ ถ้ามีแสดงว่ามีการติดตั้งโปรแกรม Sendmail ไว้แล้วและจะต้องมีการเปิดพอร์ตสำหรับ SMTP ไว้เป็นเบอร์ 25 พร้อมทั้งมีโพรเซส sendmail ทำงานอยู่ด้วย

2. ติดตั้งโปรแกรม Sendmail : ในกรณีตรวจสอบพบว่าไม่มีโปรแกรม Sendmailเราสามารถติดตั้งเองได้ โดยการดาวน์โหลดโปรแกรม Sendmail จากเว็บไซต์ ไม่ว่าจะอยู่ในรูปของแพ็คเกจ หรือเป็นโค้ดต้นฉบับก็ตาม สำหรับเว็บไซต์ที่เข้าไปดาวน์โหลดได้ดังตัวอย่าง

http://www.sendmail.org : เป็นเว็บไซต์ที่ให้ข้อมูลเกี่ยวกับโปรแกรม Sendmail รวมถึงดาวน์โหลดโปรแกรม Sendmail วิ่งจะอยู่ในรูปของโค้ดต้นฉบับ

ftp://ftp.redhat.com/pub/redhat/linux/7.2/en/os/i386/SRPMS : สำหรับลีนุกซ์ค่ายเรดแฮด สำหรับลีนุกซ์ค่ายอื่น ๆ เราสามารถเข้าไปดาวนีโหลดจากเว็บไซต์ของค่ายนั้น ๆ ได้ เช่น ftp://ftp.slackware.com/pub/slackware/slackware-current/slanckware/n1

ftp://ftp.debian.com/debian/dists/potato/main/source.mail/ เป็นต้น

ติดตั้งแบบแพ็คเกจ

ในกรณีที่เราดาวน์โหลดโปรแกรม Sendmail ที่เป็นแบบแพ็คเกจ (*.rpm) เราสามารถใช้คำสั่ง rpm ช่วยในการติดตั้งได้ โดยเรียกคำสั่ง

rpm –iv –force sendmail-cf-8.11.6-2

ติดตั้งจากโค้ดต้นฉบับ

สำหรับโปรแกรม Sendmail ที่อยู่ในรูปของโค้ดต้นฉบับ เราแบ่งได้เป็น 2 แบบด้วยกันคือ

1. โค้ดต้นฉบับที่อยู่ในรูปของแพ็คเกจ เช่น sendmail-8.11.6.-2.7.1.src.rpm โค้ดแบบแพ็คเกจเราต้องใช้คำสั่ง

rpm -I sendmail-8.11.6-3.src.rpm"

ทำการแตกเอาโค้ดต้นฉบับมาก่อน ซึ่งเมื่อใช้คำสั่ง rpm ติดตั้งแล้ว โค้ดต้นบับจะอยู่ที่ไดเร็คทอรี /usr/src/redhat/SOURCES/

ไดเร็คทอรีนี้มีไฟล์ที่จำเป็นต้องใช้ในระบบเมล เช่น

*.patch	เป็น patch สำหรับโปรแกรมต่าง ๆ
sendmail.init	เป็นไฟล์ที่ใช้เรียกหรือหยุดระบบเมล เรานำไปใช้โดยการสำเนาไปไว้ที่ /etc/init.d/ และเปลี่ยนชื่อเป็น sendmail
sendmail.pam กับ sendmail.conf	เป็นไฟล์ที่ใช้เกี่ยวกับ pam (Pluggable Authentication Modules)
sendmail-redhat.mc	เป็นไฟล์ที่เรานำมาคอมไพล์เพื่อให้ได้ไฟล์ configuration sendmail.cf
sendmail.sysconfig	เป็นไฟล์ที่ใช้กำหนดระยะเวลาในการส่งเมลในคิวใหม่ เรานำไปใช้โดยสำเนาไปที่ /etc/sysconfig/ เปลี่ยนชื่อเป็น sendmail ไฟล์นี้จะถูกเรียกใช้จาก /etc/init.d/ sendmail ตอนระบบเมลเริ่มทำงาน
aliases	ไฟล์ aliases เรานำไปใช้โดยสำเนาไปไว้ที่ /etc

sendmail-8.11.6 เป็นไดเร็คทอรีที่เป็นโค้ดต้นฉบับ เมื่อเราเข้าไปคอมไพล์จะได้ไฟล์โปรแกรมต่าง ๆ เช่น sendmail, rmail, vacation เป็นต้น ซึ่งเราจะพูดถึงขั้นตอนการคอมไพล์ และติดตั้งในหัวข้อถัดไป

ที่ไดเร็คทรอรี /usr/src/redhat/SPECS/ จะมีไฟล์ชื่อ sendmail.spec ซึ่งไฟล์นี้เราสามารถนำไปสร้างกลับให้เป็นแพ็คเจจของโค้ดต้รฉบับได้ โดยเรียกคำสั่ง "rpm -ba sendmail.spec"

ผลลัพธ์ที่ได้จะเป็นแพ็คเกจโค้ดต้นฉบับอยู่ที่ไดเร็คทอรี /usr/src/redhat/SPPMS/ โดยไฟล์ที่ต้องการให้อยู่ในแพ็คเกจเราสามรถกำหนดใน sendmail.spec ได้

2. โค้ดต้นฉบับที่อยู่ในรูปของไฟล์ tar เช่น sendmail.8.12.3.tar.gz ถ้าเป็นโค้ดที่อยู่ในรูปของไฟล์ tar เราต้องทำการแตกไฟล์ tar ออกมาก่อน โดยเรียกคำสั่ง "tar xvfz sendmail.8.12.3.tar.gz" จะได้ไดเร็คทอรีของโค้ดต้นฉบับอยู่ที่ไดเร็คทอรีปัจจุบัน

ก่อนการติดตั้งโปรแกรมจากโค้ดต้นฉบับ เราควรอ่านไฟล์ README ก่อนสำหรับรายละเอียดเบื้องต้น และในที่นี้ควรอ่านไฟล์ INSTALL ด้วยเพื่อเป็นแนวทางปฏิบัติในการติดตั้งโปรแกรมสำหรับ

3. ให้เปลี่ยนไดเร็คทอรีเข้าไปยังไดเร็คทอรีของ sendmail-8.12.3 ที่เราแตกจากไฟล์ tar ดังตัวอย่าง ในรูปที่ 11-8 แล้วเรียกคำสั่งดังนี้ในการคอมไพล์

[root@redhat sendmail-8.12.3]# sh Build

เมื่อคอมไพล์เสร็จแล้วจะมีไดเร็คทอรี obj.* ขึ้นมา จากตัวอย่างจะได้ไดเร็คทอรี obj.Linux.2.4.7-10.i586 โดยภายใต้ไดเร็คทอรีนี้ จะประกอบไปด้วยไดเร็คทอรีย่อยของโปรแกรมต่าง ๆ ที่คอมไพล์เสร็จแล้ว

ถ้าเราเปลี่ยนไดเร็คทอรีเข้าไปในไดเร็คทอรีย่อย ๆ นี้ จะพบว่ามีไฟล์เต็มไปหมดทั้งไฟล์ต้นฉบับ (*.c) ไฟล์ออบเจกต์ (*.o) แต่จะมีไฟล์หนึ่งที่เป็นไฟล์ไบนารี่ ซึ่งพร้อมนำไปใช้งานได้

4. ทำการติดตั้งไบนารี่ไฟต่าง ๆ ที่คอมไพล์แล้ว ไปไว้ยังไดเร็คทอรีที่กำหนดไว้ โดยเราต้องไปอยู่ ณ ไดเร็คทอรีเหมือนตอนที่เราทำการคอมไพล์ แล้วเรียกคำสั่งต่อไปนี้

[root@redhat sendmail-8.12.3]# make install

ไฟล์ที่นำไปติดตั้งยังไดเร็คทอรีที่กำหนดมีดังนี้

/usr/sbin/ มีไฟล์ editmap, mailstats, makemap, praliases, smrsh

/usr/sbin/ ไฟล์ vacation

สำหรับไฟล์โปรแกรม sendmail จะไม่ถูกนำไปติดตั้ง ซึ่งเราอาจใช้วิธีสำเนาไปไว้ที่ /usr/sbin/ เองก็ได้ และควรทำลิงค์ไฟล์ไปยังไดเร็คทอรี /usr/sbin/ ด้วย ดังคำสั่งต่อไปนี้

[root@redhat root]# In -s /usr/sbin/sendmail /usr/sbin/sendmail

5. ทำการคอมไพล์ และติดตั้งไฟล์ configuration คือ sendmail.cf และ submit.cf ให้เราเปลี่ยนไดเร็คทอรีเข้าไปยังไดเร็คทอรีย่อย cf/cf จะพบว่ามีไฟล์ที่ใช้ในการคอมไพล์เป็นไฟล์ configuration

ในที่นี้เราใช้ระบบปฏิบัติการเป็นลีนุกซ์ ดังนั้นควรเลือกไฟล์ที่เตรียมไว้สำหรับลีนุกซ์ นั่นคือไฟล์ geneic-linux.mc โดยเรียกคำสั่งดังต่อไปนี้

[root@redhat cf]# make install-cf CF=generic-linux

สำหรับ CF= ให้ใส่ชื่อไฟล์ที่เป็นชนิด *.mc

เมื่อคำสั่งทำงานเสร็จแล้วจะพบว่ามีไฟล์ /etc/mail/sendmail.cf และ /etc/mail/submit.cf เกิดขึ้น ให้เราทำลิงค์ไปไว้ที่ /etc/ ด้วย โดยเรียกคำสั่งดังนี้

[root@redhat sendmail]# In -s /etc/mail/sendmail.cf /etc/sendmail.cf

[root@redhat sendmail]# In -s /etc/mail/submit.cf /etc/submit.cf

6. ติดตั้งไฟล์โปรแกรม sendmail และคู่มือ โดยให้เราเปลี่ยนไดเร็คทอรีออกมายังไดเร็คทอรีย่อย sendmail ที่ไม่ได้อยู่ในไดเร็คทอรี obj.* ให้ลองอ่านไฟล์ชื่อ SECURITY ซึ่งจะบอกว่าเราควรสร้างบัญชีผู้ใช้ชื่อ "smmsp" และกลุ่มผู้ใช้ "smmsp" โดยมีหมายเลขกลุ่มเป็น 25 หลังจากนั้นจึงค่อยเรียกคำสั่งดังต่อไปนี้

[root@redhat sendmail]# sh Build install

คำสั่งนี้จะทำการสำเนาโปรแกรม sendmail ไปไว้ที่ /usr/sbin/ และคู่มือไปไว้ที่ /usr/man/man5/,/usr/man/man8/ เป็นต้น ที่ไดเร็คทอรียังมีไฟล์ชื่อ "aliases" ซึ่งเป็นไฟล์ที่เราจำเป็นต้องใช้สำหรับติดตั้งเมลเซิร์ฟเวอร์ ให้เราทำสำเนาไปไว้ที่ /ect/mail และทำลิงค์จาก /etc ด้วย โดยเรียกคำสั่ง

[root@redhat sendmail]# in -s /etc/mail/aliases /etc/aliases

7. ทำสำเนาไฟล์ภายใต้ไดเร็คทอรีย่อย sendmail-8.12.3/cf/ ทั้งหมดไปไว้ยังไดเร็คทอรี /usr/share/sendmail-cf ด้วยคำสั่งต่อไปนี้

[root@redhat cf]# find . –depth -print l cpio -pmdv /usr/share/sendmail-cf

ภายใต้ไดเร็คทอรีนี้เราอาจไม่ค่อยได้ใช้เท่าไรนัก แต่ว่าเมื่อไรที่ต้องการปรับเปลี่ยนไฟล์ sendmail.cf หรือต้องการเพิ่มคุณสมบัติให้กับ sendmail เราจะได้ใช้ประโยชน์จากไฟล์ต่าง ๆ เหล่านี้

สำหรับไฟล์ sendmail.cf ที่คอมไพล์จากไฟล์ generic-linux.mc จะเป็นไฟล์ configuration ที่แบบมาตรฐานทั่วไป ยังไม่มีการเพิ่มคุณสมบัติอื่นเข้าไป

ถ้าเราต้องการให้ระบบเมลของเรามีคุณสมบัติพิเศษเพิ่ม เช่น ป้องกันการรีเลย์สแปมเราสามารถแก้ไขไฟล์ *.mc โดยเพิ่ม FEATURE (feature_name, argument) เช่น

FEATURE('access_db','hash -o /etc/mail/access')dnl

เป็นต้น ซึ่งเราสามารถเข้าไปอ่านรายละเอียดเกี่ยวกับคุณสมบัติต่าง ๆ ได้ที่ /usr/share/sendmail-cf/feature/

ตัวอย่าง ไฟล์ sendmail.mc ที่เพิ่มคุณสมบัติ

```
divert(-1)
divert(0)dnl
VERSIONID('$Id: generic-linux.mc,v 8.1 1999/09/24 22:48:05 gshapior Exp $')
OSTYPE(linux)dnl
DOMAIN(generic)dnl
FEATURE('mailertable','hash -o /etc/mail/mailertable')dnl
FEATURE(virtusertable','hash -o /etc/mail/ virtusertable')dnl
FEATURE('access_db','hash -o /etc/mail/ access')dnl
FEATURE('blacklist_recipients')dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

ไฟล์ต่าง ๆ ที่เกี่ยวข้องกับระบบเมล

หลังจากติดตั้งโปรแกรม sendmail ซึ่งทำหน้าที่เป็น MTA แล้ว ก่อนเข้าสู่การติดตั้งเมลเราควรรู้จักไฟล์ต่าง ๆ ที่เกี่ยวข้องกับระบบเมลกันก่อนว่า ไฟล์ไหนทำหน้าที่อะไรและมีรูปแบบในการปรับค่าแบบไหน เพื่อที่เราจะสามารถติดตั้งระบบเมลให้ได้ตามจุดประสงค์ที่เราต้องการที่สุด ดังรายละเอียดต่อไปนี้

ไฟล์ /etc/sendmail.cf

เป็นไฟล์ configuration ที่ถูกอ่านทุกครั้งที่ปรแกรม sendmail ถูกเรียกขึ้นมาทำงาน เพื่อให้ปรแกรม sendmail ทำงานตามข้อกำหนดในแฟ้มนี้ เช่น กำหนดว่าจะใช้เครื่องไหนเป็นรีเลย์โฮสต์ จะให้มี

การเก็บเมลที่ไม่ได้ส่งไปไว้ที่เซิร์ฟเวอร์กี่วัน หรือกำหนดขนาดของเมลที่ส่งไว้ไม่ให้เกิดกี่เมกกะไบต์ก็ได้ เป็นต้น

ไฟล์นี้แบ่งได้ 2 ส่วนใหญ่ ๆ ด้วยกันคือ

1. ส่วนของการกำหนดค่า (Macro) : ในส่วนนี้เป็นการกำหนดค่าให้กับมาโคร ซึ่งจะถูกนำไปใช้ในตอนที่โปรแกรม sendmail เข้ามาทำงานตามกฎ (rule set) ที่อยู่ในส่วนท้ายของไฟล์ sendmail.cf เราสามารถกำหนดค่าให้กับมาครได้ 3 แบบดังนี้

1.1 กำหนดมาโครแบบค่าเดียว (Define Macro) : การกำหนดแบบนี้เราสามารถกำหนดให้มาโครมีค่าได้เพียงค่าเดียวเท่านั้น ซึ่งมีรูปแบบดังต่อไปนี้

D+ อักษร 1 ตัว + ค่าที่กำหนดให้ (กำหนดมาโครในกรณีมีชื่อเป็นตัวอกษรตัวเดียว)

D+{อักษร > 1 ตัว} + ค่าที่กำหนด (กำหนดมาโครในกรณีที่มีชื่อหลายตัว)

ตัวอย่าง การกำหนดค่ามาโคร

Djzoo.co.th (กำหนดมาโคร "j" ให้มีค่าเป็น zoo.co.th)

D{domain_name}zoo.co.th (กำหนดมาโคร "{domain_name}" ให้มีค่าเป็น zoo.co.th)

1.2 กำหนดมาโครแบบหลายค่า (Class Macro) : เป็นการกำหนดค่าให้กับมาโครในกรณีที่ต้องการให้มีค่าหายค่า ซึ่งมีรูปแบบดังต่อไปนี้

C+ อักษร 1 ตัว + ค่าที่ 1 ค่าที่ 2 (กำหนดมาโครในกรณีที่มีชื่อเป็นตัวอักษรตัวเดียว)

C+{อักษร > 1 ตัว} +ค่าที่ 1 ค่าที่ 2 (กำหนดมาดครในกรณีที่มีชื่อหลายตัวอักษร

ตัวอย่าง การกำหนดค่ามาโคร

Cwlocalhost redbat redbat.zoo.co.th

C{receive_domain}localhost redhat redhat.zoo.co.th

1.3 กำหนดมาโครแบบหลายค่าในรูปแบบไฟล์ (File Class Macro) : รูปแบบนี้เป็นการกำหนดค่าให้มาโครแบบหลาย ๆ ค่า เหมือนกับแบบ class macro ต่างกันตรงที่ค่าที่กำหนดจะอยู่ในรูปแบบของไฟล์ แทนที่เราจะกำหนดให้อยู่ในไฟล์ sendmail.cf ซึ่งมีรูปแบบดังต่อไปนี้

F+ อักษร 1 ตัว + ชื่อไฟล์ (กำหนดมาโครในกรณีที่มีชื่อเป็นอักษรตัวเดียว)

F+{อักษร > 1 ตัว} + ชื่อไฟล์ (กำหนดมาโครในกรณีมีชื่อหลายตัวอักษร)

ตัวอย่าง การกำหนดค่ามาโคร

Fw/etc/mail/local-host-name (sendmail จะเข้าอ่านค่าของมาโครจากไฟล์ที่กำหนด)

F(receive_domain_file)/etc/mail/local-host-name

ในการกำหนดค่าไม่ว่าจะใช้คำสั่ง D, C, F ก็ตามจะพิมพ์ให้ติดกับขอบซ้ายไม่มีการเว้นด้วยช่องว่าง

เช่น

l Cwlocahost redhat **(แบบนี้ผิด)**

lCwlocahost redhat **(แบบนี้ถูก)**

หมายเหตุ : "l" ในที่นี้สมมติว่าเป็นขอบซ้ายของไฟล์

2. ส่วนกฎของ (Rule Set) : ในส่วนนี้เป็นเสมือนกับเป็นโค้ดโปรแกรมของระบบเมล โดยเมื่อมีการรับ-ส่งเมล จะต้องผ่านกระบวนการตามกฎก่อน ซึ่งถ้าเราเข้าใจการทำงานในกกต่าง ๆ เราสามรถสร้างกฎของเราเองเพิ่มเข้าไปในไฟล์ sendmail.cf ได้ เช่น เราอาจสร้างกฎเพิ่มให้ตรวจสอบดูว่าเมลฉบับไหนใหญ่เกินไป ให้รออยู่ในคิวก่อน จนกระทั่งตอนกลางคืนค่อยนำคิวเหล่านั้นมาส่ง เป็นต้น

ไฟล์ /etc/aliases

เป็นไฟล์ที่ใช้กำหนดนามแฝงของผู้ใช้แต่ละคนหรือแบบกลุ่มผู้ใช้ก็ได้ เมื่อมีเมลผ่านเข้ามาที่พอร์ต SMTA sendmail จะทำการตรวจสอบว่าเป็นเมลฉบับนั้นจะให้ส่งไปหาผู้รับที่อยู่ภายในเครื่องเมลเซิร์ฟเวอร์หรือว่าที่อื่น ถ้าเป็นการส่งภายในเมลเซิร์ฟเวอร์ จะทำการอ่านไฟล์ aliases ว่าที่ของผู้รับนั้นจะส่งผ่านไปยังกล่องจดหมายของผู้รับคนไหน หรือผู้รับกลุ่มไหน รูปแบบการกำหนดนามแฝงในไฟล์ aliases เป็นดังนี้

นามแฝง : **email-address1 [,email-address2,]**

ตัวอย่าง ผู้รับมีบัญชีบนเมลเซิร์ฟเวอร์เป็น my0022 มี e-mail address เป็น tony_w@mailyou.com ดังนั้นเมื่อเมลมาถึงนาย tony ใน /etc/aliases ไฟล์จะต้องกำหนดดังนี้

Tony_w: my0022

ซึ่ง my0022 จะเป็นกล่องจดหมายของผู้ใช้ด้วย ในกรณีที่ต้องการกำหนดให้มีการส่งตัวไปยังที่อื่นด้วยเราสามารถเพิ่มใน aliaese ได้ดังนี้

Tony_w: my0022, tony_w@yahoo.com

เนื่องจากเราสามารถกำหนดให้มีการส่งเมลไปหาผู้รับหลาย ๆ คนพร้อมกันได้ ดังนั้นเราก็สามารถสร้างนามแฝงกลุ่มผู้รับได้ เช่น ต้องการสร้างนามแฝงกลุ่มเพื่อส่งหาผู้จัดการทุกคน โดยให้ email-address เป็น all_mgr@mailyou.com

All_mgr: somsak_s, sompong_p, somchai_c, summate_m, somyos_y

โดยสมาชิกในนามแฝงกลุ่มก็จะส่งต่อไปยังกล่องจดหมายของแต่ละคนดังนี้

somsak_s: my0203

sompong_p: my0522

somchai_c: my2122

summate_m: my0302

somyos_y: my2205

เมื่อมีการแก้ไขไฟล์ aliases จะต้องทำการอัพเดตให้ระบบรู้โดยเรียงคำสั่งดังนี้

 # newaliases

นามแฝงก็คือ ชื่อผู้รับที่อยู่หน้าเครื่องหมาย @ นั่นเอง เมื่อ sendmail เข้ามาค้นหาในไฟล์ aliases จะนำเฉพาะชื่อที่อยู่หน้า @ ของ e-mail address เข้าตรวจสอบเท่านั้น

ไฟล์ /etc/mail/access

เป็นคุณสมบัติที่เราใช้กำหนดรีเลย์ โดยเรากำหนดว่าเครื่องไหนบ้าง โดเมนไหนบ้าง IP ไหนที่ยอมให้มีการรีเลย์ผ่านเครื่องเราได้ หรือเครื่องไหนบ้าง โดเมนไหนบ้างที่เราไม่ยอมให้รีเลย์ผ่านเครื่องเราได้เลย เราสามารถใช้ไฟล์นี้ในการป้องกันการรีเลย์สแปมได้ นั่นคือเราสามารถป้องกันผู้ไม่ประสงค์ดีนำเครื่องเมลเซิร์ฟเวอร์เราไปสแปมเครื่องคนอื่นอีกทีได้ ดังตัวอย่างต่อไปนี้

192.168.1.1 RELAY (ยอมให้ IP นี้รีเลย์ผ่านเครื่องเราได้)

203.25.141 RELAY (ยอมให้เน็ตเวิร์คนี้รีเลย์ผ่านเครื่องเราได้)

gooddmain.com RELAY (ยอมให้โดเมนนี้รีเลย์ผ่านเครื่องเราได้)

badomain.com 550 Mail from your domain is denied(ไม่รับเมลจากโดเมนนี้พร้อมแสดง ข้อความตอบกลับ)

203.203.203.2550 Internet mail may not be sent from this host.(ไม่รับเมลจาก IP นี้ พร้อมแสดงข้อแสดงข้อความตอบกลับ)

Badguy@somedomain.com REJECT (ปฏิเสธเมลที่มาจากที่อยู่นี้)

เมื่อมีการแก้ไขจะต้องทำการอัพเดตให้ระบบรู้เช่นกัน โดยเรียกคำสั่งดังนี้

 # makemap hash /etc/mail/access

ไฟล์ /etc/mail/mailertable

เป็นคุณสมบัติที่เราใช้ในการกำหนด mailer ให้กับโดเมนที่ต้องการส่งไป โดยมีรูปแบบดังนี้

Domain Mailer:domain

หรือ

Domain Mailer:host.domain

หรือ

Domain Mailer:[IP]

ดังตัวอย่าง

Somedomain.co.th smtp:redhat.zoo.co.th (เมื่อมีการส่งเมลถึงโดเมน Somedomain.co.th

Somedomain.net relay:[203.144.12.22] (เมื่อมีการส่งเมลถึง Somedomain.net กำหนดให้ใช้ mailer relay ส่งไปยังเครื่องที่มี IP 203.144.12.22)

เมื่อมีการแก้ไขจะต้องทำการอัพเดตให้ระบบรู้เช่นกัน โดยเรียกคำสั่งดังนี้

 # makemap hash /etc/mail/mailertable < /etc/mail/mailertable

ไฟล์ /etc/mail/virtusertable

เป็นคุณสมบัติที่เราใช้ในการกำหนดว่า เมื่อมีการส่งเมลออกไปหาผู้รับที่โดเมนอื่น ให้ทำการส่งไปยังผู้รับที่อยู่บนเครื่อง หรือส่งไปยังผู้รับที่เรากำหนด ดังตัวอย่าง

Someone@domain.com my2017 (เมื่อมีการส่งเมลถึงผู้รับ Someone@domain.com เมลนั้นจะส่งไปหาผู้รับ my2017 ที่อยู่บนเครื่องแทน)

@abc.com one@elsewhere.com (เมื่อมีการส่งเมลถึงผู้รับใดก้ตามที่อยู่โดเมน abc.com ให้ส่งไปหาผู้รับ one@elsewhere.com แทน)

ถ้าเราต้องการให้ชื่อผู้รับเหมือนเดิมให้เรากำหนดเป็น %1 ดังนี้

@abc.com %1@elsewhere.com

ถ้ามีการส่งเมลถึงผู้รับ man@abc.com เมลนั้นจะถูกส่งไปที่ man@elsewhere.com แทนเมื่อมีการแก้ไขจะต้องทำการอัพเดตให้ระบบรู้เช่นกัน โดยเรียกคำสั่งนี้

 # makemap hash/etc/mail/virtusertable < /etc/mail/virtusertable

ไฟล์ /etc/sysconfig/sendmail

ไฟล์นี้จะถูกอ่าน เมื่อมีการเรียกไฟล์ /etc/init.d/sendmail เพื่อเรียกระบบเมลขึ้นมา ภายในไฟล์จะเป็นการกำหนดตัวแปร 2 ตัวดังนี้

DAEMON=yes yes= กำหนดให้ sendmail ทำงาน daemon

QUEUE=1h 1h= กำหนดให้มีการเมลที่ยังส่งไม่ออกในคิวทุก ๆ ชั่วโมง

ไฟล์ /usr/sbin/ sendmail

เป็นไฟล์โปรแกรม sendmail ที่ทำหน้าที่เป็นตัวรับส่งเมล

ไฟล์ /var/log/maillog

เป็นไฟล์ที่บันทึกข้อความทุกอย่างที่เกี่ยวข้องกับระบบเมล ไม่ว่าจะเป็นวันเวลาของเมลเข้า-ออก ใครเป็นคนส่ง แล้วส่งหาใครบ้าง รีเลย์เครื่องไหนเป็นต้น

ไฟล์ /usr/share/ sendmail-cf/

เป็นไดเร็คทอรีที่เก็บไฟล์ configuration ที่จำเป็นในการสร้างไฟล์มาโคร *.mc หรือ sendmail.cf

ไฟล์ /var/spool/mqueue/

เป็นไดเร็คทอรีที่เก็บเมลที่รอการส่ง โดยเมลตและฉบับจะประกอบด้วยไฟล์ 2 ไฟล์คือ **ไฟล์ที่** เก็บ header โดยชื่อไฟล์จะขึ้นต้นด้วย "qxxx" ส่วนอีกไฟล์คือ ไฟล์ที่เก็บเนื้อเมล ชื่อไฟล์จะขึ้นต้นด้วย "dxxx" โดย xxx ของเมลแต่ละฉบับจะไม่ซ้ำกัน ดังรูปที่ 11-11 และตัวอย่างไฟล์ที่เป็น header กับ เนื้อเมล

ตัวอย่าง ไฟล์ header เมล qfg3F6pYM01738

V4

T1018853494

K1018853534

N1

P50

L3/66/252627

Fb

$_root@redhat

RPFD:what_happen@hotmail.com

H?P?Return-Path: g>

H??Received:(from root@localhost)

By redhat (8. 11.6/8. 11.6) id g3F6pyMo1738

For what_happen@hotmail.com; Mon, 15 Apr 2002 13:51:34 +0700

H?D?Date: Mon, 15 Apr 2002 13:51:34 +0700

H?F?From: root <root>

H?x?Full-Name:root

H?M?Message-Id: 200204150651.g3F6pYMo1738@redhat

H??To: what_happen@hotmail.com

K??Subject: first mail.

ตัวอย่าง ไฟล์เนื้อเมล dfg3F6pYMo1738

HI

This is the contain in this mail.

You can see them when you got the mail.

สำหรับการตรวจสอบว่ามีเมลที่อยู่ในคิวกี่เมล และแต่ละเมลใครเป็นคนส่ง ส่งหาใครบ้าง ขนาดเท่าไร เราตรวจสอบได้โดยเรียกคำสั่ง mailq

โดยมีรายละเอียดดังนี้

Q-ID หมายเลขคิวของเมล

Size ขนาดของเมล

Q-Time เวลาที่ทำการส่งเมล

Sander/Recipient แสดง email address ผู้ส่งและผู้รับ โดย email ที่อยู่บรรทัดแรกคือผู้ส่ง ส่วนบรรทัดถัดมาจะเป็นผู้รับทั้งหมด และในกรณีที่ส่งไม่ได้จะมีข้อความแสดงให้เห็นดังรูป

จะพบว่ามีบางบรรทัดที่เครื่องหมาย * ต่อท้ายหมายเลขคิวที่คอลัมน์แรก หมายถึง เมลที่กำลังดำเนินการส่งอยู่นั่นเอง

ในบางครั้งพบว่ามีเมลค้างอยู่ในคิวเป็นจำนวนมาก เราอาจใช้คำสั่งดังต่อไปนี้ เพื่อจัดการเคลียร์เมลที่ค้างให้ส่งออกไปให้หมด

 # /usr/sbin/sendmail -q

เนื่องจากเมลที่ค้างอยู่อาจเกิดจากเมลที่กำลังส่งอยู่ หรือเมลที่ส่งครั้งไม่ได้ภายในเวลาที่กำหนดก็จะต้องรออยู่ในคิวจนกว่าจะได้เวลาเรียกส่งใหม่อีกครั้ง ถ้าสมมติเรากำหนดให้เวลาในการส่งใหม่นาน และมีเมลเข้ามามากก็จะทำให้เมลที่ค้างอยู่ในคิวเยอะไปด้วย

การกำหนดเวลาการเรียกเมลในคิวมาส่งใหม่ กำหนดได้ที่ไฟล์ /etc/sysconfig/sendmail ที่ตัวแปร QUEUE

เริ่มติดตั้งเมลเซิร์ฟเวอร์

การติดตั้งเมลเซิร์ฟเวอร์เราแยกการติดตั้งออกเป็น 2 ส่วนด้วยกันคือ

การติดตั้งในส่วนของระบบปฏิบัติการ

ระบบเมลก็ต้องทำงานสัมพันธ์กับระบบปฏิบัติการ ดังนั้นจึงจำเป็นต้องปรับค่าบางอย่างให้รองรับกับการทำงานของระบบเมลด้วย ดังขั้นตอนต่อไปนี้

ไฟล์ /etc/hosts : ระบบจะมีการตรวจสอบไฟล์ /etc/hosts ดูว่ามีการกำหนดชื่อเครื่อง โดยระบุโดเมนให้ด้วยหรือเปล่า ถ้ายังไม่กำหนดเมื่อมีการเรียกระบบเมลขึ้นมาทำงาน จะมีข้อความแสดงขึ้นพร้อมทั้งบันทึกลงในไฟล์ /var/log/mailog ด้วยดังนี้

My unqualified host name (hostname) unknown; sleeping for retry

ดังนั้นเราจึงต้องกำหนดนามแฝงพร้อมทั้งระบุโดเมนด้วย ดังตัวอย่าง

203.148.11.22 redhat.zoo.co.th

ไฟล์ /etc/nsswitch.conf : ในกรณีที่เรามีการส่งเมลไปยังโดเมนอื่นด้วย เราจำเป็นต้องกำหนดให้ระบบค้นหาโดเมนนั้นจาก DNS (Domain Name System) ถ้าไม่เช่นนั้นแล้วเมลจะไม่สามารถส่งออกไปได้ เพราะว่าเมลเวิร์ฟเวอร์ไม่รู้จักเมลเซิร์ฟเวอร์หรือรีเลย์โฮสต์ปลายทางนั่นเอง เราสามารถกำหนดได้โดยเพิ่มคำ dns ดังนี้

Hosts: files dns

ไฟล์ /etc/resolv.conf : เมื่อมีการกำหนดว่าจะใช้ DNS ดังนั้นเราจำเป็นต้องติดตั้งระบบ DNS แบบ Client เพื่อให้เมลเซิร์ฟเวอร์รู้ว่า ถ้าต้องการค้นหาโดเมนไหน ให้เราเข้าไปถามเครื่องที่เป็นเนมเซิร์ฟเวอร์ไหนได้บ้าง ดังตัวอย่างดังต่อไปนี้

Search zoo.co.th

Nameserver 203.148.11.20

Nameserver 203.1480.22.45

การติดตั้งในส่วนของระบบเมล

เมื่อส่วนในระบบปฏิบัติการพร้อมแล้ว ก็มาเข้าสู่การติดตั้งในส่วนของเมลกัน ดังขั้นตอนต่อไปนี้

ไฟล์ /etc/เมื่อมีการกำหนดว่าจะใช้ DNS ดังนั้นเราจำเป็นต้องติดตั้งระบบ DNS แบบ Client เพื่อให้เมลเซิร์ฟเวอร์รู้ว่า /etc/sendmail.cf : โปรแกรม sendmail ในเวอร์ชันหลัง ๆ ที่ออกมา ได้จัดเตรียมให้เราสามารถติดตั้งได้ง่ายขึ้น ดังต่อไปนี้

กำหนดโมเมนที่เมลเซิรีฟเวอร์จะรับ : ในส่วนนี้เป็นการกำหนดว่าเมลที่ส่งเข้ามายังเครื่องเมลเซิร์ฟเวอร์ของเรามีโดเมน หรือโฮสต์เนมอะไรบ้างที่เราจะรับ ดังนี้

Cwlocahost redhat redhat.zo.co.th

 # file containing name of hosts for which we receive email

#Fw/etc/mail/local-host-names

จากตัวอย่าง ให้เราเลือกก่อนว่าจะกำหนดแบบคลาสมาโคร หรือว่าไฟล์มาโคร ถ้ากำหนดเป็นคลาสมาโคร เราต้องใช้เครื่องหมาย # ยกเลิกการใช้แบบไฟล์คลาสมาโครด้วย เมื่อเรกำหนดค่าตามข้างต้นนั่นหมายความว่า เราจะรับเมลที่ส่งเข้ามาเป็น *@redhat.zoo.co.th, *@zoo.co.th

กำหนดโดเมน : ในกรณีที่ sendmail ไม่กำหนดโดเมนให้ เราสามรถกำหนดให้กับมาโคร j ได้ ซึ่งมาโครนี้โดยปกติจะมีค่าเป็น "ชื่อเครื่อง.ชื่อโดเมน" ในบางครั้งถ้าเราต้องการที่อยู่เมลที่ส่งออกไปไม่ให้มีชื่อเครื่องติดไปด้วย เราสามารถกำหนดที่มาโครนี้เช่นกัน ดังนี้

#my official domain name

 # ... define this only if sendmail cannot automatically determine your domain

#Dj$w.Foo.COM

Djzoo.co.th

กำหนดหนดรีเลย์โฮสต์ : ในกรณีที่เมลเซิร์เวอร์เราไม่สามารถติดต่อกับภายนอกได้โดยตรง เราจำเป็นที่ต้องกำหนดรีเลย์โฮสต์ให้ เพื่อบอกว่าเมลทุกฉบับที่ส่งออกให้ส่งไปยังรีเลย์โฮสต์ส่งต่ออีกที แต่ถ้าเมลเซิร์ฟเวอร์สามารถรัส่งได้ด้วยตัวเอง เราก็ไม่ต้องกำหนดก็ได้ ดังนี้

 # *Smart" relay host (may be mull)

DSzoorelay

การกำหนดออปชัน : ในบางครั้งเพื่อให้ระบบเมลทำงานดียิ่งขึ้น เราจำเป็นต้องเปลี่ยนแปลงค่าออปชันที่ระบบเมลกำหดมาให้ ซึ่งเราจะรู้ได้ไงว่าค่าไหนเป็นค่าออปชัน ดังตัวอย่าง

```
##################
#    Option      #
##################
.
# maximum message size
#O MaxMessageSize=1000000    (กำหนดขนาดของเมลที่จะส่งได้(Byte)


#queue directory
O QueueDirectory=/var/spool/mqueue    (ไดเร็คทอรีที่เก็บเมลที่รอส่ง)


# timeouts (many of these)
O Timeout.queueretrueturn=5d      (กำนดเวลาก่อนที่จะตีกลับถึงผู้ส่ง)
O Timeout.queueretrueturn=4h      (กำหนดเวลาในการบอกผู้ส่งว่ายังส่งไม่ได้)
```

ในกรณีที่เราใช้โปรแกรม sendmail ที่ลงแบบแพ็คเกจติดต่อกับเครื่องเราผ่านทางพอร์ต smtp ได้ ให้ให้เราแก้ในไฟล์ sendmail.cf ที่บรรทัดข้างล่างนี้แล้วทำการหยุดระบบ แล้วรียกระบบเมลขึ้นใหม่

 # SMTP daemon options

DaemonPortOptions=Port=smtp.Addr=1277.0.0.1,Name=MTA

แก้เป็น

DaemonPortOptions=Port=smtp,Addr=0.0.0.0, Name=MTA

ไฟล์ /etc/aliases : เมลเซิร์ฟเวอร์เป็นเครื่องที่มีผู้ใช้รวมถึงกล่องจดหมายเก็บอยู่ ดังนั้นเราจำเป็นต้องสร้างนามแฝง เพื่อให้เมลที่ส่งเข้ามาสามารถไปถึงกล่องจดหมายของผู้รับได้อย่างถูกต้อง ดังได้กล่าวถึงรูปแบบไว้แล้วข้างต้น

เริ่มการใช้งานระบบเมล

หลังจากติดตั้งเสร็จแล้ว เราสามารถเรียกระบบเมลให้พร้อมทำงาน หรือว่าหยุดการทำงานได้ด้วยการเรียกสคริปต์ดังนี้

 # /etc/init.d/sendmail [stop | start | restart]

เราสามารถตรวจสอบได้ว่าตอนนี้ระบบเมลพร้อมทำงานแล้วหรือยังโดยการ

 # telnet localhost 25

ทดสอบการส่งเมล

การทดสอบเราจะใช้โปรแกรม mail ที่ระบบปฏิบัติการเตรียมพร้อมไว้อยู่แล้ว โดยใช้ออปชั่น "-v" เพิ่มเพื่อดูแต่ละขั้นตอนการส่ง เราเรียกคำสั่ง mail เพื่อส่งเมลได้ดังนี้

 # mail –v <recipient e-mail address>

การติดตั้งระบบเมลเพื่อรับเมลจากภายนอกโดเมน

เมื่อติดตั้งเมลเซิร์ฟเวอร์แล้ว การจะติดต่อรับเมลจากภายนอกโดเมนได้ จำเป็นต้องมีการปรับค่าในระบบ DNS เพื่อให้โดเมนอื่นรู้จักว่า เมื่อส่งเมลมาที่โดเมนของเราจะต้องติดต่อที่เครื่องไหน นั่นคือการกำหนด Mail Exchange ให้กับโดเมน

ไฟล์ที่เราต้องเข้าไปแก้ไขจะเป็นไฟล์ zone domain ลองตรวจสอบดูที่ไฟล์ /etc/named.conf อีกทีว่าไฟล์นั้นติดตั้งอยู่ที่ไหน และใช้ชื่อว่าอะไร โดยเราจะเพิ่ม resource record MX เข้าไปดังตัวอย่าง

ตัวอย่าง zone domain, zone.zoo.co.th

```
$TTL    86400
@       IN      SOA     whale.zoo.co.th  root.whale.zoo.co.th     (
                                2002021601              ;       Serial
                                                8H              ; Refresh
                                                3H              ; Retry
                                                1W              ; Expire
                                                1D              ;Minimum TTL
                IN      NS              whale.zoo.co.th  ; primary
                IN      NS              zebra.zoo.co.th  ;secondary
                IN      MX              10      zoorelay.zoo.co.th.
                IN      MX              20      zoorelay2.zoo.co.th.
;       host in network  203.148.11.0
...
;       host in network  203.148.22.0
...
```

หลังจากกำหนด Mail Exchange แล้ว ต้องทำการหยุดระบบ DNS แล้วเรียกขึ้นมาทำงานใหม่หลังจากนั้นเมื่อมีเมลส่งมายังโดเมนของเรา รีเลย์โฮสต์ต้นทางจะติดต่อกับเครื่องที่เป็น Mail Exchange นั่นคือ เครื่อง zoorelay.zoo.co.th หรือ zoorelay2.zoo.co.th แต่จะมีโอกาสเข้ามาที่เครื่อง zoorelay.zoo.co.th มากกว่า เนื่องจากใน DNS กำหนดให้มี Priority มากกว่า

บรรณานุกรม

[1] W.Zhang and et al. Linux Virtual Server project.
http://www.linuxvirtualserver.org/ , 1998

[2] R.S.Engelschall. Load balancing your web site:Practical approaches for distributing http traffic.
Web Technique Magazine , 315/, May 1998
http://www.webtechniques.com

[3] A.Roberson and et al. High-Availaibility linux project.
http://www.linux-ha.org/, 1998

[4] T.Brisco. Dns support for load balancing.
http://www.ietf.org/rfc/rfc1794.txt, April 1995 RFC 1794

[5] http://www.ultramonkey.org

[6] http://www.redhat.com