การจัดแบ่งภาระงานของเวปคลัสเตอร์ด้วยลำดับเสมือน

Load Balancing Systems for Web Cluster via Virtual Token

กริช กองศรีมา[1] ผศ.มยุรี เลิศเวชกุล[2]

Krit Kongsrima Mayuree Lertwatechakul

[1]มหาวิทยาลัยราชภัฏนครราชสีมา [2]สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

E-mail kkongsrima@hotmail.com , klmayure@kmitl.ac.th

บทคัดย่อ

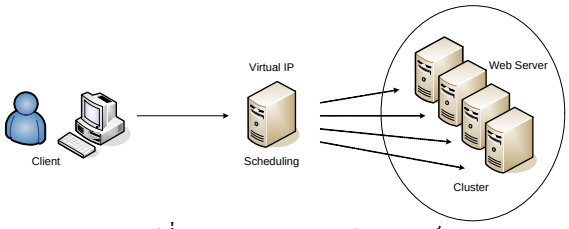
บทความฉบับนี้นำเสนอแนวทางการจัดแบ่งภาระงาน (Load Balancing) ของเวปคลัสเตอร์ โดยใช้การจัดลำดับเสมือนร่วมกับการประเมินประสิทธิภาพการให้บริการของเวปเซิร์ฟเวอร์จากทรัพยากรที่มีอยู่ขณะนั้น คือ การทำงานของซีพียู หน่วยความ จำหลัก และจำนวนการเชื่อมต่อ กล่าวคือ มีการใช้โปรโตคอลในการจับกลุ่มเพื่อทำให้เวปเซิร์ฟเวอร์จัดลำดับในการให้บริการกันเองโดยกำหนดเป็นลำดับเสมือน เวปเซิร์ฟเวอร์ลำดับแรกจะพิจารณาให้บริการถ้าหากทรัพยากรของตนมีเพียงพอ หลังจากให้บริการแล้วก็จะถูกจัดลำดับใหม่ให้อยู่ที่ลำดับท้ายสุด โดยทุกครั้งหลังจากการพิจารณาให้บริการของเวปเซิร์ฟเวอร์ลำดับแรกนั้น ไม่ว่าจะให้บริการได้หรือไม่ได้ก็จะต้องมีการจัดลำดับใหม่ให้ลำดับแรกไปเป็นลำดับสุดท้ายและลำดับรองลงไปได้ขยับลำดับความสำคัญให้มากขึ้นทีละหนึ่งลำดับ จนกระทั่งทุกเครื่องได้มาเป็นลำดับแรกและเวียนไปเป็นลำดับสุดท้ายอีกครั้งหนึ่ง ทำให้สามารถลดขั้นตอนการจัดแบ่งภาระงานที่อาจจะเป็นคอขวดของระบบเมื่อมีการร้องขอมากขึ้นเรื่อยๆได้

Abstract

This article proposes “Distributed Load Balancing in Web Clustering System via Virtual Token Technique”. The system uses Grouping Protocol as to add or delete web cluster’s members and to make a virtual schedule for them. Load balancing could be done in round-robin fashion, the current head server of the web cluster will be rescheduled as the last server either after its servicing or its service denying caused by its resources insufficiency (memory, CPU usage). The virtual token technique could be implemented by using Group Protocol and monitoring of TCP call request segments and call accept segments. The technique could reduce the need of centralized load balancing server and may avoid of bottleneck of the centralize control point at the high request situation.

1. คำนำ

เวปคลัสเตอร์ได้เป็นที่รู้จักอย่างแพร่หลายถึงประโยชน์ในการใช้งานเพื่อให้สามารถรองรับการบริการกับจำนวนของการร้องขอปริมาณมากๆได้ ซึ่งในการใช้งานนั้นจะต้องเลือกระบบปฏิบัติการและโปรแกรมจัดการเพื่อให้เกิดการทำงานของระบบคลัสเตอร์ที่จะต้องเข้ากันได้ เช่น ระบบปฏิบัติการลีนุกส์ที่สามารถใช้ร่วมกับลีนุกส์เวอร์ชวลเซิร์ฟเวอร์ (Linux Virtual Server)[1] หรือระบบปฏิบัติการวินโดวส์ที่มีเน็ทเวอร์คโหลดบาลานซ์ซิ่ง (Network Load Balancing)[2] ที่สามารถทำงานในลักษณะเป็นซอร์ฟแวร์ควบคุมการจัดแบ่งภาระงาน หรือใช้ฮาร์ดแวร์ทำหน้าที่นี้ เช่น ซิสโก้โลคอลไดเร็กเตอร์ (Cisco LocalDirector)[3] และเวปมักส์(WebMux)[4] เป็นต้น ซึ่งแนวคิดพื้นฐานของระบบเวปคลัสเตอร์นั้นจะมีเครื่องหลัก (Master Server) คอยตรวจสอบและส่งต่อเพื่อให้การร้องขอเข้ามาได้รับบริการจากเครื่องใดเครื่องหนึ่งในกลุ่มของเวปเซิร์ฟเวอร์ ซึ่งกล่าวได้ว่าเครื่องหลักนี้ทำหน้าที่เป็นตัวจัดแบ่งภาระงาน (Scheduler/Dispatcher)

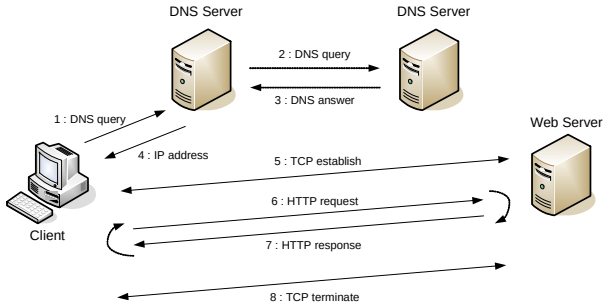


รูปที่ 1 การทำงานของเวปคลัสเตอร์

สิ่งที่ได้ศึกษาในเบื้องต้นในโครงการวิจัยนี้ คือ การเปรียบเทียบเพื่อให้ได้แนวคิดของระบบให้ได้ข้อดีที่เป็นจุดเด่นในการนำไปพัฒนาระบบต้นแบบให้ใช้งานได้จริง สามารถใช้งานในลักษณะของโอเพนซอร์สและสะดวกในการจัดการ เพื่อรองรับจำนวนของการร้องขอปริมาณมากได้

1.1 พื้นฐานการทำงานของเวปเซิร์ฟเวอร์

พื้นฐานของการร้องขอข้อมูลผ่านเวปนั้นมีขั้นตอนดังนี้ คือ



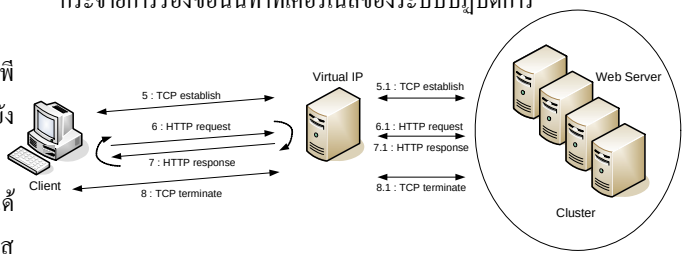
รูปที่ 2 ขั้นตอนการทำงานในการรับบริการเวปเพจ

1. เครื่องไคลเอนท์ทำการร้องขอบริการแปลงหมายเลขไอพีแอดเดรสจากชื่อโดเมนหรือชื่อเวปไซต์ที่ผู้ใช้ต้องการเชื่อมต่อไปยัง DNS ที่อยู่ภายในเครือข่าย
2. DNS ที่อยู่ภายในเครือข่ายทำการแปลงชื่อโดเมนไปเป็นไอพีแอดเดรสให้ โดยค้นหาข้อมูลเดิมที่อยู่ในแคชหรือถามไปยัง DNS ในระดับถัดไป
3. DNS ในระดับถัดไปหรือ DNS ที่ดูแลโดเมนนั้นๆ ที่ได้สอบถามไปจะทำการตอบกลับมาเป็นหมายเลขไอพีแอดเดรส
4. DNS ที่อยู่ภายในเครือข่ายจะได้หมายเลขไอพีที่ตอบกลับมานั้น และส่งคำตอบมาให้เครื่องไคลเอนท์
5. เครื่องไคลเอนท์จะทำการสร้างการเชื่อมต่อโดยโปรโตคอลทีซีพี (TCP) ไปยังเวปเซิร์ฟเวอร์ โดยอ้างการเชื่อมต่อระหว่างหมายเลขไอพีและพอร์ท ซึ่งทั้งเครื่องต้นทางและปลายทางจะทำการสร้างการเชื่อมต่อระหว่างกัน (Three-way handshake)
6. เมื่อสร้างการเชื่อมต่อเสร็จสิ้นแล้ว เครื่องไคลเอนท์จะทำการร้องขอข้อมูลเวปเพจจากเวปเซิร์ฟเวอร์โดยโปรโตคอลเอชทีทีพี (HTTP)
7. เวปเซิร์ฟเวอร์ทำการตอบกลับโดยส่งเอกสารของหน้าเวปเพจกลับไปที่ไคลเอนท์ ซึ่งในขั้นตอนนี้เครื่องไคลเอนท์จะมีการส่งข้อมูลการร้องขอและตอบกลับในลักษณะการโต้ตอบไปมาจนกระทั่งได้ข้อมูลของหน้าเวปเพจทั้งหมด

1.2 พื้นฐานการทำงานของคลัสเตอร์เวปเซิร์ฟเวอร์

การทำงานพื้นฐานบนคลัสเตอร์นั้นจะต้องประกอบด้วยการเชื่อมต่อกันด้วยเครือข่ายความเร็วสูงและส่วนของการจัดการที่ทำให้ทุกเครื่องเสมือนเป็นเครื่องเดียวกัน ซึ่งจะสามารถสร้างเป็นคลัสเตอร์แบบเปิดหรือแบบปิดก็ได้ การทำงานของเวปคลัสเตอร์นั้นแตกต่างจากการทำงานแบบปกติ คือ ขั้นตอนที่มีการรับการร้องขอการเชื่อมต่อที่เข้ามานั้นจะเข้ามาที่หมายเลขไอพีเสมือน ซึ่งจะต้องมีวิธีการกระจายการร้องขอนี้ไปยังสมาชิกที่อยู่ในกลุ่มโดยวิธีการใดวิธีการหนึ่ง โดยจะให้ผ่านเครื่องหลักหรืออาจทำโดยไม่ผ่านเครื่องหลักก็ได้ เหมือนกับเครื่องหลักคอยทำหน้าที่ชี้ให้การร้องขอที่เข้ามานั้นไปรับบริการที่เวปเซิร์ฟเวอร์เครื่องนั้นๆในกลุ่ม

จากเทคนิคในการรับการร้องขอและกระจายการร้องขอที่ได้รับเข้ามาในคลัสเตอร์นั้นนิยมใช้การอ้างหมายเลขไอพีเสมือน (Virtual IP) เพื่อให้เสมือนมีเครื่องที่มีหมายเลขนี้อยู่จริงและสามารถสร้างการเชื่อมต่อกลับไปได้ การที่ใช้หมายเลขเสมือนนี้ทำให้ต้องใช้หมายเลขของฟิสิคัล (Physical Address/MAC Address) เสมือนด้วยเช่นกัน แต่ก็อาจจะใช้เทคนิคแตกต่างกันไป เช่น ทำในลักษณะยูนิคาสต์ มัลติคาสต์หรือบรอดคาสต์ เพื่อทำให้ทุกเครื่องสามารถรับเอาการร้องขอที่เข้ามาในระบบของคลัสเตอร์นี้ไปใช้พิจารณาต่อไปได้ และในการกระจายการร้องขอนั้นทำที่เคอร์เนลของระบบปฏิบัติการ



รูปที่ 3 ขั้นตอนการทำงานในการรับบริการเวปเพจของเวปคลัสเตอร์

2. วิธีการจัดแบ่งภาระงาน

แนวทางที่สามารถทำได้เพื่อการจัดภาระงานในระบบเวปคลัสเตอร์นั้น สามารถทำได้ในชั้นใดๆ ของสถาปัตยกรรมทีซีพีไอพี[5][6][7] เช่น

- ชั้นดาต้าลิงค์ (DataLink layer)

ทำในลักษณะของการจัดภาระงานในชั้นดาต้าลิงค์ที่เป็นชั้นที่สองของสถาปัตยกรรมทีซีพีไอพีโดยพิจารณาจากข้อมูลในชั้นทรานสปอร์ต (Transport layer) เมื่อมีการร้องขอเข้ามาจากอินเทอร์เน็ตจะต้องเข้ามาที่เครื่องที่ทำหน้าที่เป็นตัวกระจายงาน (Dispatcher) เครื่องที่เป็นตัวกระจายงานจะต้องทำการจัดภาระงานโดยให้มีเป้าหมายเป็นเครื่องใดเครื่องหนึ่งจากเวปเซิร์ฟเวอร์ที่จะต้องเป็นผู้ให้บริการ เครื่องกระจายงานจะต้องทำการส่งต่อไปยังเครื่องที่ให้บริการโดยการแก้ไขข้อมูลเฟรมในส่วนของหมายเลขฟิสิคัล (Physical/MAC address) เพื่อส่งต่อไปยังเครื่องนั้น การแก้ไขข้อมูลแพ็กเก็ตนี้ทำเฉพาะที่ชั้นของดาต้าลิงค์เท่านั้นซึ่งไม่เกี่ยวข้องกับชั้นเน็ทเวอร์ค (Network layer) ดังนั้นวิธีการนี้ก็ไม่จำเป็นต้องแก้ไขข้อมูลในส่วนหัวของไอพีแพ็กเก็ต (IP Header) คือส่วนของค่าผลรวม (Checksum) เครื่องที่เป็นเวปเซิร์ฟเวอร์ในระบบ

ที่ทำงานแบบนี้จะต้องมีหมายเลขไอพีแอดเดรสเดียวกัน และเป็นหมายเลขเดียวกันกับตัวกระจายงานด้วย เครื่องที่เป็นเวปเซิร์ฟเวอร์ที่จะต้องให้บริการนั้นจะทำการตอบสนองกลับไปยังเครื่องที่ได้ร้องขอเข้ามาโดยตรง เครื่องที่ทำหน้าที่กระจายงานจะต้องทราบถึงเซสชั่นที่กำลังเชื่อมต่ออยู่โดยพิจารณาจากหมายเลขไอพีและพอร์ท เพื่อให้การเชื่อมต่อที่มีอยู่เดิมนั้นได้มีการร้องขอไปที่เครื่องเดิม เราอาจเรียกวิธีการนี้ว่าเป็นวิธีการแบ่งภาระงานโดยใช้ข้อมูลจากเซสชั่น คือ หมายเลขไอพีและพอร์ท (Session load balancing based on IP address and TCP port) วิธีนี้เป็นการทำงานร่วมกันระหว่างชั้น ดาต้าลิงค์และชั้นทรานสปอร์ต (L4/L2)

- ชั้นเน็ทเวอร์ค (Network layer)

ทำในลักษณะของการจัดภาระงานที่ชั้นเน็ทเวอร์คที่เป็นชั้นที่สามของสถาปัตยกรรมทีซีพีไอพี โดยวิธีการหาเส้นทาง (Routing) ที่จะมองว่าตัวกระจายงานเป็นเกตเวย์ (Gateway) ทำหน้าที่ในการเชื่อมต่อทั้งภายในและภายนอก ดังนั้นวิธีนี้จะแตกต่างจากการกระจายงานในชั้นดาต้าลิงค์ คือ การส่งต่อแพ็กเก็ตจะทำด้วยการแก้ไขหมายเลขของไอพีแอดเดรส การแก้ไขหมายเลขไอพีแอดเดรสนี้ทำให้ต้องมีการแก้ไขค่าผลรวมในข้อมูลส่วนหัวของไอพีด้วย ในส่วนของการตอบกลับนั้นจะต้องมีการทำงานผ่านตัวกระจายงานที่ทำหน้าที่เป็นเกตเวย์ ซึ่งวิธีการกระจายภาระงานแบบนี้ทำให้ตัวกระจายงานต้องทำงานหนักกว่าแบบแรกที่ได้กล่าวมาข้างต้น

- ชั้นแอพพลิเคชั่น (Application layer)

โดยทำในลักษณะของการแบ่งภาระงานจากข้อมูลภายในชั้นแอพพลิเคชั่นจากชั้นที่ห้าของสถาปัตยกรรมทีซีพีไอพี เช่น ข้อมูลของเครื่อง (Content routing based on Host tag) ข้อมูลจากที่อยู่ (Entire URL) ข้อมูลจากคุกกี้ (Dynamic cookie location) หรือข้อมูลจากนามสกุลของเอกสาร (File extension)

หากแบ่งการจัดภาระงานให้เป็นสองแบบ คือ คงที่ตามวิธีการที่กำหนด (Static Scheduling) กับวิธีที่สามารถปรับเปลี่ยนได้ตามสภาวะแวดล้อมในการทำงาน (Dynamic Scheduling) ซึ่งทั้งสองแบบนี้สามารถจัดภาระงานได้โดยวิธีการต่างๆ คือ

- คงที่ตามวิธีการที่กำหนด (Static Scheduling)

เช่น วนรอบเพื่อให้บริการ (Round Robin) อัตราส่วนในการให้บริการ (Ratio) สุ่มเพื่อเลือกผู้ให้บริการ (Random)

- ปรับเปลี่ยนได้ตามสภาวะแวดล้อมในการทำงาน (Dynamic Scheduling)

เช่น พิจารณาจากเวลาตอบสนอง (Response time) พิจารณาจากจำนวนการเชื่อมต่อ (Least Connection) พิจารณาจากการใช้งานซีพียู (CPU Load) พิจารณาจากการใช้งานหน่วยความจำ (Memory Utilization)

2.1 วิธีการจัดแบ่งภาระงานของลีนุกส์[1]

การจัดแบ่งภาระงานของลีนุกส์เวอร์ชวลเซิร์ฟเวอร์นั้นมีวิธีการพื้นฐานอยู่ 3 กลุ่มหลักๆ คือ

- การวนรอบ (Round-Robin)

ทำงานแบบวนรอบจากลำดับแรกถึงลำดับท้ายสุดของแต่ละรอบ และสามารถกำหนดน้ำหนักในการวนรอบ (Weighted Round-Robin) ที่มีการกำหนดน้ำหนักของการรับบริการ เช่น หากเครื่องแรกมีน้ำหนักเป็น 1 และเครื่องถัดไปมีน้ำหนักเป็น 4 ดังนั้นเครื่องแรกจะต้องรองรับ 1 การเชื่อมต่อ ในขณะที่เครื่องถัดไปนั้นรองรับ 4 การเชื่อมต่อในแต่ละรอบ

- การพิจารณาจากจำนวนการเชื่อมต่อ (Least-Connection)

ทำงานในลักษณะของการพิจารณาจากความสามารถในการให้บริการจากแต่ละเครื่องมากขึ้น โดยเครื่องหลักจะคอยตรวจสอบเครื่องเวปเซิร์ฟเวอร์แต่ละตัวว่ามีจำนวนการเชื่อมต่อเท่าใด หากมีการเชื่อมต่ออยู่มากก็หมายความถึงทรัพยากรในการให้บริการมีเหลืออยู่น้อย และเพิ่มการกำหนดน้ำหนัก (Weight Least-Connection) คล้ายแบบที่กล่าวข้างต้น ทำให้กำหนดน้ำหนักของการรับบริการได้ สามารถทำงานได้ในแบบเฉพาะบริเวณ (Locality-Based Least-Connection) และแบบเฉพาะบริเวณที่สามารถโอนย้ายการให้บริการได้ (Locality-Based Least-Connection with Replication)

- การพิจารณาจากทิศทาง

คือ พิจารณาจากปลายทาง (Destination Hash Scheduling) ออกแบบมาเพื่อใช้กับ proxy-cache server cluster และพิจารณาจากต้นทาง (Source Hash Scheduling) ถูกออกแบบมาเพื่อใช้กับ LVS routers และ firewalls หลายตัว

2.2 วิธีการจัดแบ่งภาระงานของวินโดวส์[2]

ส่วนการจัดแบ่งภาระงานของเน็ทเวอร์คโหลดบาลานซ์ซึ่งมีวิธีการพื้นฐานอยู่ 3 วิธี คือ

- พิจารณาแบบไม่มีความสัมพันธ์กัน (None Affinity)

เป็นการพิจารณาจากทั้งหมายเลขไอพีและพอร์ทของเครื่องต้นทาง

- พิจารณาแบบเจาะจงความสัมพันธ์ (Single Affinity)

เป็นการพิจารณาเฉพาะหมายเลขไอพีของเครื่องต้นทาง หากเป็นไอพีเดิมก็ให้ไปที่เครื่องเดิมที่ให้บริการอยู่

- พิจารณาแบบความสัมพันธ์ของหมายเลขเครือข่ายแบบคลาส C (Class C Affinity)

เป็นการพิจารณาจากหมายเลขไอพีของเครื่องต้นทาง หากมาจากต้นทางในเครือข่ายใด ก็ให้ไปที่เครื่องเดิมที่ให้บริการนั้นทั้งหมด คล้ายกับการพิจารณาเป็นเสมือนหมายเลขเครือข่ายของคลาส C หนึ่งคลาส

2.3 เปรียบเทียบวิธีการจัดแบ่งภาระงาน

โดยรายละเอียดในการทำงานนั้นหากพิจารณาจากลีนุกส์เวอร์ชวลเซิร์ฟเวอร์ จะมีการใช้เครื่องหลักเพื่อเป็นเครื่องที่คอยจัดแบ่งภาระงาน ซึ่งการร้องขอใดๆ ที่เข้ามานั้นจะต้องมาที่เครื่องหลักนี้ก่อนเสมอ แล้วเครื่องหลักจะพิจารณาจากเงื่อนไขที่ได้กล่าวมาแล้วจากข้างต้นว่าสมควรจะส่งการร้องขอนี้ให้กับเครื่องใดอีกครั้งหนึ่ง หากพิจารณาถึงเน็ทเวอร์คโหลดบาลานซ์ซึ่งนั้น การทำงานจะอิสระจากเครื่องหลักมากกว่า เนื่องจากการพิจารณานั้นแต่ละเครื่องจะสามารถตัดสินได้เองโดยเงื่อนไขที่ได้กล่าวมาแล้ว แต่อย่างไรก็ตามทั้งสองระบบนี้ต่างก็มีข้อดีข้อเสียที่สามารถสรุปได้ คือ

- ลีนุกส์เวอร์ชวลเซิร์ฟเวอร์

มีข้อดีที่เสถียรภาพของระบบและความนิยมในการใช้งานมีเงื่อนไขหลายๆ แบบที่พิจารณาจากความสามารถในการให้บริการของแต่ละเครื่องประกอบกัน ข้อเสีย คือ ต้องอาศัยเครื่องหลักในการตัดสินการให้บริการ

- เน็ทเวอร์คโหลดบาลานซ์ซิ่ง

มีข้อดีที่ความเป็นอิสระจากเครื่องหลัก ข้อเสีย คือ การพิจารณาในการให้บริการไม่ได้เน้นที่ความสามารถของเครื่องให้บริการ แต่ไปเน้นที่เครื่องที่ร้องขอเข้ามา

ดังนั้นบทความในงานวิจัยฉบับนี้จึงขอสรุปแนวทางที่เป็นไปได้ เพื่อที่จะทำให้การจัดแบ่งภาระงานที่เอาข้อดีของทั้งสองระบบนี้มาใช้งานจากแนวคิดที่ได้กล่าวมา คือ เอาข้อดีของการวนรอบ การตรวจสอบในการให้บริการจากทรัพยากรที่มีอยู่ การลดการทำงานบางส่วนหรือบางขั้นตอน ความสะดวกในการใช้งาน ความคุ้มค่าต่อการใช้งานในช่วงระยะเวลาหนึ่ง และได้พัฒนาโปรแกรมเพื่อให้สามารถทำงานตามแนวคิดนี้ เพื่อนำไปทำงานบนกลุ่มของเวปเซิร์ฟเวอร์

3. ระบบที่ออกแบบ

ระบบที่ออกแบบนั้นพัฒนาโดยภาษาซีชาร์ป (Csharp) ซึ่งเป็นส่วนหนึ่งในวิชวลสตูดิโอดอทเนท (Visual Studio .NET) ของบริษัทไมโครซอร์ฟ ทำงานร่วมกับวินโดว์โปรโตคอลแคปเจอร์ (WinPcap) ผ่านไลบรารีของชาร์ปพีแคป (SharpPcap) เพื่อทำการดักแพกเก็ตข้อมูลมาตรวจสอบการร้องขอการเชื่อมต่อและมีการสื่อสารกันระหว่างเครื่องที่เป็นสมาชิกเพื่อแจ้งสภาวะของภาระโหลดโดยพิจารณาจากการทำงานของซีพียู หน่วยความจำหลัก และจำนวนการเชื่อมต่อ ซึ่งรายละเอียดของภาระโหลดได้มาจาก

- kernel32.dll

เพื่อนำข้อมูลในส่วนของการทำงานของซีพียูมาใช้

$$CpuUsage = \frac{(SystemTime + IdleTime)\times 100}{SystemTime} \quad (1)$$

$$SystemTime = KernelTime + UserTime \quad (2)$$

- Windows Management Instrumentation (WMI)

เพื่อนำข้อมูลในส่วนของการใช้งานหน่วยความจำมาใช้

$$MemUsage = \frac{CurrentUse\times 100}{MemTotal} \quad (3)$$

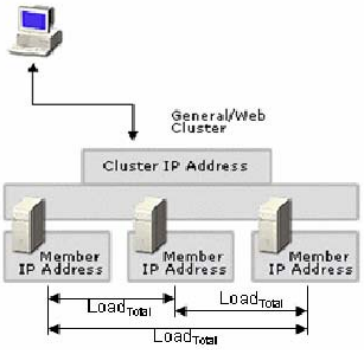
- TcpListener Class

เพื่อนำข้อมูลในส่วนของการร้องขอการสร้างการเชื่อมต่อมาใช้ ซึ่งพิจารณาจากหมายเลขไอพีและพอร์ทของการร้องขอ คือ ConnNum

สมการที่ใช้ในการพิจารณาภาระงานของแต่ละเครื่อง คือ

$$Load_{total} = \frac{ConnNum}{(CpuUsage + MemUsage)} \quad (4)$$

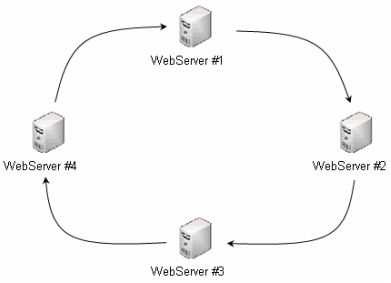
เครื่องที่เป็นสมาชิกในเวปคลัสเตอร์จะมีการแจ้งสถานะของภาระงานถึงกันเพื่อบอกระดับของภาระงาน โดยต้องการให้ทราบถึงค่าระดับภาระงานเพื่ออ้างอิง สมการที่ใช้จึงเป็นสมการง่ายๆที่ไม่เน้นถึงการบอกรายละเอียดและความถูกต้องแม่นยำมากนัก และต้นแบบในการทำงานของระบบมีส่วนประกอบต่างๆ ดังนี้



รูปที่ 4 การทำงานของระบบ

3.1 การจับกลุ่ม

การจับกลุ่มนั้นทำโดยใช้โปรโตคอลในการจับกลุ่มที่ทำงานอัตโนมัติ เพื่อให้เกิดลำดับก่อนหลังในการทำงานในแต่ละรอบ โดยให้เครื่องที่มีหมายเลขของฟิสิคัลน้อยที่สุดเป็นเครื่องลำดับแรกและหมายเลขของฟิสิคัลมากที่สุดเป็นเครื่องลำดับสุดท้าย ดังนั้นการจับกลุ่มนี้จะทำให้ข้อจำกัดของจำนวนเครื่องในคลัสเตอร์หมดไป

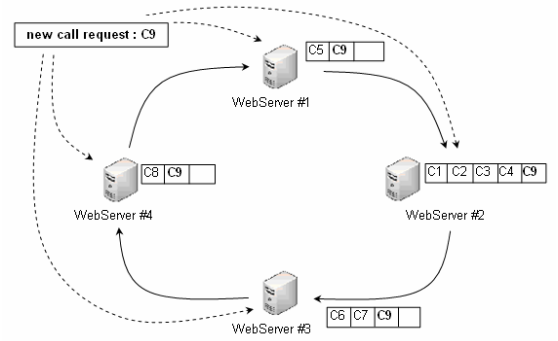


รูปที่ 5 การจับกลุ่มของสมาชิกในเวปคลัสเตอร์

3.2 การกระจายการร้องขอและการจัดแบ่งภาระงาน

ระบบที่ออกแบบนั้น ใช้วิธีการกระจายการร้องขอแบบเดียวกับเวอร์ชวลไอพีและลำดับแบบเสมือน เพื่อให้ทุกเครื่องได้รับการร้องขอที่เข้ามาทั้งหมดซึ่งไม่ต้องกระจายการร้องขออีกครั้งหนึ่ง ดังนั้นจะทำงานในลักษณะของลำดับการให้บริการ (Queue) ที่รองรับการร้องขอที่เข้ามา

- เครื่องที่มีลำดับแรกนั้นจะทำการตัดสินในการรับบริการก่อน หากเครื่องแรกรับบริการนั้นไปแล้ว เครื่องอื่นๆ ก็ต้องลบการร้องขอนั้นออกจากลำดับการให้บริการ เมื่อให้บริการไปแล้วเครื่องแรกก็จะปรับลำดับให้ไปเป็นลำดับสุดท้าย และลำดับรองลงไปก็จะปรับลำดับความสำคัญให้มากขึ้นทีละ 1 ลำดับ เพื่อให้ทุกเครื่องได้มีสิทธิ์ได้มาเป็นลำดับแรกในการให้บริการ



รูปที่ 6 การร้องขอที่เข้ามาในลำดับการให้บริการ

- หากเครื่องแรกไม่สามารถให้บริการได้ ก็จะปรับลำดับให้ไปเป็นลำดับสุดท้ายเช่นกัน ดังนั้นเครื่องที่มีลำดับที่สองจะต้องมาเป็นลำดับแรกแทนเพื่อทำการตัดสินในการรับบริการ
- การพิจารณาในการให้บริการนั้นจะพิจารณาจากการใช้งานซีพียู การใช้งานหน่วยความจำ จำนวนการเชื่อมต่อของแต่ละเครื่องเอง ซึ่งแนวคิดนี้มาจากแนวคิดของการอาสาหรือเสนอตัวในการทำงาน โดยไม่มีการสั่งให้ทำงานจากเครื่องหลัก

3.3 การสร้างการเชื่อมต่อและตอบกลับ

เมื่อทราบว่าใครเป็นผู้ที่จะต้องรับการให้บริการนั้นๆ แล้ว ก็จะเกิดการเชื่อมต่อโดยตรงกลับไปที่เครื่องต้นทาง ซึ่งแต่ละเครื่องที่ให้บริการนั้นจะต้องมีการตรวจสอบการร้องขอที่ได้รับเข้ามาอยู่เสมอ เพื่อคอยแยกระหว่างเครื่องที่มีการเชื่อมต่ออยู่เดิมและเครื่องที่ร้องขอเข้ามาใหม่ เครื่องที่มีการเชื่อมต่ออยู่เดิมนั้นก็จะถูกส่งไปที่เครื่องเดิมที่ให้บริการนั้น

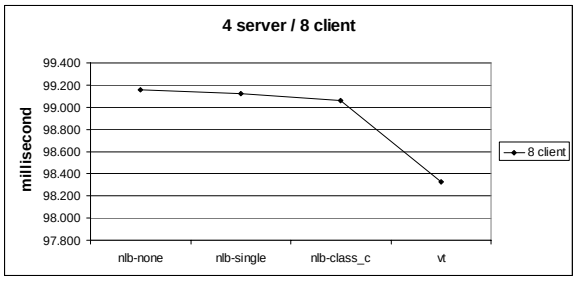
4. ผลการทดลอง

การวัดประสิทธิภาพของระบบนั้นจะพิจารณาในเรื่อง จำนวนการเชื่อมต่อ (requests/second) ปริมาณข้อมูลที่รับส่งได้ (bytes/second) และค่าเวลาเฉลี่ยที่ใช้ในการสร้างการเชื่อมต่อ (connection time) การวัดประสิทธิภาพทำบนการเชื่อมต่อเครือข่ายซึ่งมีรายละเอียดคือ เครื่องที่เป็นเซิร์ฟเวอร์มีจำนวนตั้งแต่ 1 ถึง 4 เครื่องเชื่อมต่อกันเป็นเครือข่ายผ่านอุปกรณ์สวิชชิ่งของ Allied Telesyn รุ่น AT-8024 ทำงานที่ความเร็ว 100 Mbps และมีเครื่องไคลเอนท์ 2 เครื่องเชื่อมต่อกันเป็นเครือข่ายผ่านอุปกรณ์สวิชชิ่งของ Allied Telesyn รุ่น AT-8024 ทำงานที่ความเร็ว 100 Mbps เช่นกัน อุปกรณ์สวิชชิ่งเชื่อมต่อกันด้วยความเร็ว 100 Mbps เครื่องเซิร์ฟเวอร์เป็นเครื่องแบบเดสทอป(Desktop) มีหน่วยประมวลผลแบบ Intel Pentium4 ความเร็ว 2.8 GHz ติดตั้งระบบปฏิบัติการวินโดว์ 2003 เอ็นเทอร์ไพรส์ (Windows Server 2003 Enterprise edition) เซิร์ฟเวอร์เครื่องที่หนึ่งและสองมีหน่วยความจำหลักเครื่องละ 512 MB เซิร์ฟเวอร์เครื่องที่สามและสี่มีหน่วยความจำหลักเครื่องละ 256 MB ใช้เวปเซอร์ฟเวอร์ของวินโดว์ คือ อินเทอร์เน็ทอินฟอร์เมชันเซิร์ฟเวอร์ (Internet Information Server : IIS 6.0) ที่ทำงานร่วมกับระบบการจัดภาระงานที่ออกแบบคือเวอร์ชวลโทเคน (Virtual token : vt) เทียบกับการจัดภาระงานของเน็ทเวอร์คโหลดบาลานซ์ซึ่ง (Network load balancing : nlb) แบบไม่มีความสัมพันธ์กัน (None Affinity)

สาเหตุที่เปรียบเทียบระหว่างการทำงานแบบเวอร์ชวลโทเคน (Virtual token : vt) กับการจัดภาระงานของเน็ทเวอร์คโหลดบาลานซ์ซึ่ง (Network load balancing : nlb) แบบไม่มีความสัมพันธ์ (None Affinity) คือจากการทดสอบการจัดภาระงานทั้ง 3 แบบของเน็ทเวอร์คโหลดบาลานซ์ซึ่งนั้น โดยการจำลองเครื่องไคลเอนท์จำนวน 8 ไคลเอนท์ให้ร้องขอไปที่เวปเซิร์ฟเวอร์จำนวน 4 เครื่องที่มีการจัดภาระงานในแต่ละแบบทั้งไคลเอนท์และเซิร์ฟเวอร์เชื่อมต่อกันในอุปกรณ์สวิชชิ่งตัวเดียวกันของ Allied Telesyn รุ่น AT-8024 ทำงานที่ความเร็ว 100 Mbps พบว่าทั้ง 3 แบบนั้นสามารถรองรับการเชื่อมต่อและมีเวลาเฉลี่ยที่ใช้ในการเชื่อมต่อใกล้เคียงกัน และการพิจารณาภาระงานแบบไม่มีความสัมพันธ์นั้นก็จะพิจารณาจากทั้งหมายเลขไอพีแอดเดรสและหมายเลขพอร์ทเช่นเดียวกับเวอร์ชวลโทเคน จึงเลือกเอามาเป็นตัวหลักในการเปรียบเทียบ

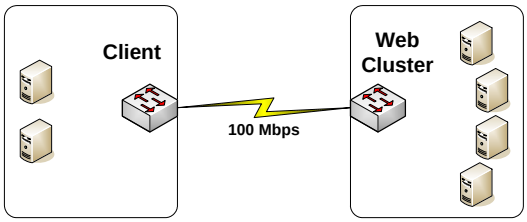
Detail	nlb-none	nlb-single	nlb-class_c	vt
requests per second (req/s)	29.009	29.116	28.957	26.291
average connection time (ms)	99.156	99.120	99.061	98.324

ตารางที่ 1 เวลาในการสร้างการเชื่อมต่อโดยเฉลี่ย
และปริมาณการร้องขอ ของ 4 server / 8 client



รูปที่ 7 กราฟปริมาณการร้องขอและเวลาในการเชื่อมต่อ

เครื่องไคลเอนท์สองเครื่องเป็นเครื่องคอมพิวเตอร์แบบพกพา (Notebook) ไคลเอนท์เครื่องแรกมีหน่วยประมวลผลแบบ Intel PentiumM ความเร็ว 1.5 GHz มีหน่วยความจำ 736 MB เครื่องที่สองมีหน่วยประมวลผลแบบ Intel Celeron ความเร็ว 1.3 GHz มีหน่วยความจำ 512 MB ทำการจำลองการร้องขอจาก 1 จนถึง 16 ไคลเอนท์ เพิ่มขึ้นทีละ 2 เท่า โดยใช้โปรแกรมเวปเบนซ์ (WebBench 5.0)[8] เป็นเครื่องมือในการทดสอบ จากการทดสอบโดยให้มีการร้องขอโดยตรงจากไคลเอนท์ไปยังเวปเซิร์ฟเวอร์ผ่านทางหมายเลขไอพีเสมือน



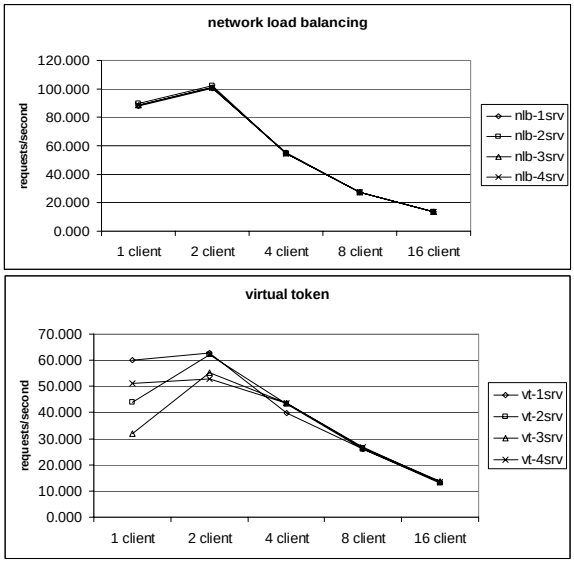
รูปที่ 8 แผนผังการเชื่อมต่อเครือข่าย

จากผลที่ได้พบว่าระบบที่ออกแบบ (virtual token) มีปริมาณการร้องขอและสามารถสร้างการเชื่อมต่อในกรณีที่มีเครื่องไคลเอนท์จำนวนน้อยเครื่องได้ไม่ดีเท่าเน็ทเวอร์คโหลดบาลานซ์ซิ่ง (network load balancing) แต่เมื่อจำนวนไคลเอนท์มีมากขึ้นนั้นจะมีความสามารถในการรองรับได้ใกล้เคียงกัน

client	single-srv	nlb-1srv	nlb-2srv	nlb-3srv	nlb-4srv
1 client	90.079	88.200	89.667	88.421	88.521
2 client	100.156	100.544	101.883	101.188	101.017
4 client	52.628	55.035	54.403	54.526	54.443
8 client	26.609	27.231	27.245	27.405	27.354
16 client	13.351	13.627	13.598	13.666	13.650

client	vt-1srv	vt-2srv	vt-3srv	vt-4srv
1 client	60.058	43.771	31.908	50.958
2 client	62.729	62.017	55.350	52.942
4 client	39.803	43.378	43.556	43.656
8 client	25.990	26.168	26.406	26.852
16 client	13.226	13.163	13.564	13.511

ตารางที่ 2 ปริมาณการร้องขอ (requests/second)



รูปที่ 9 กราฟปริมาณการร้องขอ

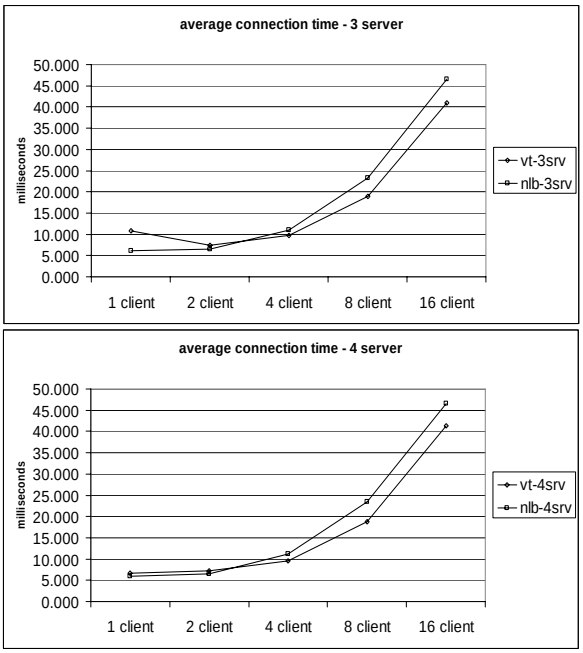
จากปริมาณการร้องขอที่สามารถให้บริการได้นี้ส่งผลต่อปริมาณข้อมูลที่รับส่งได้ (bytes/second) ซึ่งเป็นสัดส่วนโดยตรงต่อกันในความสามารถในการร้องขอข้อมูลและรับส่งข้อมูล

client	single-srv	nlb-1srv	nlb-2srv	nlb-3srv	nlb-4srv
1 client	6.000	6.060	6.053	6.082	6.027
2 client	6.672	6.605	6.546	6.543	6.561
4 client	11.600	10.986	11.079	11.069	11.177
8 client	24.376	23.609	23.492	23.357	23.420
16 client	48.254	46.780	46.878	46.649	46.545

client	vt-1srv	vt-2srv	vt-3srv	vt-4srv
1 client	6.001	8.317	10.851	6.723
2 client	6.532	6.584	7.390	7.219
4 client	11.109	10.090	9.692	9.486
8 client	19.179	18.974	18.923	18.719
16 client	41.982	42.713	41.008	41.353

ตารางที่ 3 เวลาในการสร้างการเชื่อมต่อโดยเฉลี่ย
(average connection time)

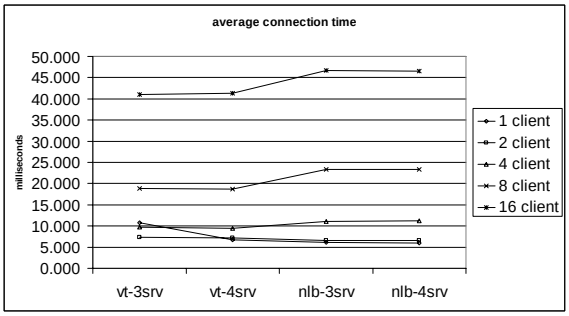
จากผลการการทดลองพบว่าระบบที่ออกแบบนั้นมีแนวโน้มว่าหากมีเครื่องไคลเอนท์มากขึ้น เมื่อเทียบเวลาในการสร้างการเชื่อมต่อในระบบที่ออกแบบจะใช้เวลาน้อยกว่าเน็ทเวอร์คโหลดบาลานซ์ซิ่ง ดังการเปรียบเทียบที่จำนวนเวปเซิร์ฟเวอร์ 3 เครื่องและ 4 เครื่อง ซึ่งในการนำไปใช้งานจริงนั้นจะมีทั้งเซิร์ฟเวอร์และไคลเอนท์ที่มากกว่าการทดสอบจึงคาดการณ์ว่าหากมีการร้องขอเข้ามามากขึ้นเรื่อยๆเวลาที่ใช้ในการจัดภาระงานแบบลำดับเสมือนเพื่อรับการร้องขอจะยิ่งสูงขึ้น แต่ก็ยังน้อยกว่าการทำงานของเน็ทเวอร์คโหลดบาลานซ์ซิ่ง



รูปที่ 10 กราฟเวลาในการสร้างการเชื่อมต่อโดยเฉลี่ย

5. สรุป

การจัดภาระงานแบบลำดับเสมือนที่ออกแบบนั้นมีความสามารถในการลดเวลาในการเชื่อมต่อ (connection time) สามารถเร่งขั้นตอนของการสร้างการเชื่อมต่อได้ดีขึ้นกว่าแบบพื้นฐาน คือการทำงานของเน็ทเวอร์คโหลดบาลานซ์ซึ่งทั้ง 3 แบบ เนื่องจากการจัดภาระงานนั้นจะต้องมีกระบวนการในการพิจารณาถึงเครื่องที่เหมาะสมในการรับภาระงาน ดังนั้นเวลาที่ใช้พิจารณาในส่วนนี้จึงหน่วงในการจัดภาระงานพอสมควร การจัดภาระงานแบบลำดับเสมือนช่วยให้การเชื่อมต่อเร็วขึ้นได้เพราะกระจายไปยังเป้าหมายในลักษณะตามลำดับของผู้ให้บริการแล้วจึงพิจารณาการรับภาระงานทีหลัง ดังนั้นความเร็วที่ได้จึงมาจากจุดนี้ พบว่าเวลาที่ใช้ในการเชื่อมต่อเร็วขึ้น 11 เปอร์เซ็นต์ เมื่อเทียบกับการจัดภาระงานแบบไม่มีความสัมพันธ์ จากผลของการทดลองวัดประสิทธิภาพเพื่อยืนยันถึงความสามารถของระบบต้นแบบดังแนวคิดนั้น ผู้วิจัยจะนำแนวทางที่ได้นี้ไปแก้ไขและพัฒนาระบบการทำงานที่สามารถใช้งานบนระบบปฏิบัติการอื่นๆ ได้จริงต่อไป



รูปที่ 11 กราฟแสดงเวลาในการสร้างการเชื่อมต่อโดยเฉลี่ย

6. กิตติกรรมประกาศ

ขอขอบพระคุณโปรแกรมวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏนครราชสีมา ที่เอื้อเฟื้ออุปกรณ์การทดลองที่ใช้ในงานวิจัยนี้

เอกสารอ้างอิง

[1] Red Hat Cluster Suite , http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/cluster-suite/

[2] How Network Load Balancing Works , http://www.microsoft.com

[3] Cisco LocalDirector , http://www.cisco.com/univercd/cc/td/doc/product/iaabu/localdir/index.htm

[4] WebMux product page , http://www.redhillnetworks.com/products/spec.htm

[5] V.Cardellini, E.Casalicchio, M. Colajanni and P.S. Yu, “The State of the Art in Locally Distributed Web-Server Systems”, ACM Computing Surveys, Vol.34, No.2, June. 2002, pp. 263-311.

[6] T.Schroeder, S.Goddard and B.Ramamurthy, “Scalable Web server clustering technologies”, IEEE Network,Vol.14, No.3, May/June. 2000, pp. 38-45.

[7] V.Cardellini, M. Colajanni and P.S. Yu, “Dynamic Load Balancing on Web-Server Systems”, IEEE Internet Computing, Vol.3, No.3, May/June. 1999, pp. 28-39.

[8] WebBench , http://cs.uccs.edu/~cs526/webbench/webbench.htm