การเปรียบเทียบสมรรถนะการกระจายภาระงานระหว่าง NGINX และ Apache mod_proxy สำหรับการให้บริการเว็บ

พนมพร มีเสงี่ยม[1]
ชัยพร เขมะภาตะพันธ์[2]

บทคัดย่อ

งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบคุณสมบัติและประสิทธิภาพการทำงานการกระจายภาระงานระหว่าง NGINX และ Apache mod_proxy ที่ใช้เป็นเครื่องแม่ข่ายพร็อกซีและโหลดบาลานซ์เพื่อกระจายภาระงานให้กับ web server หรือ web application ที่อยู่ด้านหลังต่อไป ซึ่งซอฟต์แวร์ทั้ง 2 นี้เป็น open-source และเป็นที่นิยมอย่างแพร่หลายในประเทศไทย การทดลองจะใช้วิธีการจำลอง โดยกำหนดให้มีผู้ใช้งานจำนวน 1000 session ร้องขอข้อมูลผ่านเว็บ web พร้อมกัน แล้วจึงทำการวิเคราะห์ผลการทดลองที่ได้จากความสามารถในการรับภาระงาน ระยะเวลาที่ใช้ในการตอบสนอง การใช้ทรัพยากรของเครื่องคอมพิวเตอร์ ได้แก่ การใช้งานหน่วยประมวลผล และหน่วยความจำ ผลการศึกษาพบว่าซอฟต์แวร์ทั้ง 2 มีความสามารถด้านคุณสมบัติการทำงานที่คล้ายกัน สามารถใช้งานได้ทั้ง HTTP/1.1 และ HTTP/2 ได้ทั้งคู่ แต่มีความแตกต่างกันด้านวิธีการกระจายภาระงานที่มีกระบวนการต่างกัน ส่วนการเปรียบเทียบด้านประสิทธิภาพพบว่า NGINX มีประสิทธิภาพเกือบทุกด้านดีกว่า Apache mod_proxy โดยสามารถรองรับ session โดยรวมได้สูงกว่า Apache mod_proxy ประมาณ 15.6% อย่างไรก็ตาม Apache mod_proxy มีชุดโปรแกรมเสริมความสามารถมากกว่า ทำให้เพิ่มเติมฟังก์ชันการทำงานต่าง ๆ เช่น การรักษาความมั่นคงปลอดภัยได้ครบถ้วนกว่าเมื่อใช้ Apache ในขณะที่ NGINX ทำได้ยากกว่ามาก

Abstract

This research aims to compare features and performances between NGINX and Apache mod_proxy used as a proxy and load balance server for distribution traffics to behind web servers or web application systems. Both NGINX and Apache mod_proxy are an open-source software and are widely used in Thailand. The experiments achieved by using simulations that each simulation generated 1000 sessions to request data on web servers via

---

[1]นักศึกษาหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์
[2] อาจารย์ที่ปรึกษา

a studied proxy server simultaneously. Then, performance analyses based on load performances, response time, and resource utilization including CPU usage and memory consumption will be achieved. Studied results found that both software has a similar features and support HTTP/1.1 and HTTP/2. However, for the load balance feature, NGINX can provide overall performance better than Apache mod_proxy. NGINX can support request sessions about 15.6% higher than Apache mod_proxy. However, Apache mod_proxy has more features to bundle with, for example, security features can be easily completed with Apache while NGINX is harder to implement.

1. บทนำ

ในการออกแบบ Web server โดยทั่วไปมีแนวทางการออกแบบให้สามารถรองรับจำนวนผู้ใช้งานได้จำนวนหนึ่งและรองรับการขยายได้ในระยะเวลา 1-3 ปี ทั้งนี้การออกแบบ Server นั้นจะไม่ออกแบบให้ Web server เป็น Server ที่มีขนาดใหญ่เนื่องจากจะต้องใช้เงินลงทุนกับ Hardware ที่สูงเกินกว่าความจำเป็น และประกอบกับ Hardware มีการพัฒนาอย่างต่อเนื่องและมีผลิตภัณฑ์ใหม่ๆออกมาอย่างรวดเร็ว แต่หากผู้ใช้งาน Website หรือ Web Application มีปริมาณความต้องการ การใช้งานเพิ่มขึ้นและ Web server เดิมที่มี ไม่สามารถจัดหาอุปกรณ์เพื่อมา Upgrade ให้สามารถใช้งานได้เพียงพอต่อความต้องการ จำเป็นจะต้องมีวิธีการที่จะเข้ามาช่วยให้ Web server สามารถรองรับการให้บริการได้เพียงพอต่อความต้องการของผู้ใช้งาน อีกทั้งระบบที่ดีควรจะต้องสามารถรองรับการเสียหายของ Web server ได้อย่างน้อย 1 เครื่อง หรือ N+1 [1]

Load Balance เป็น Technology ที่ออกแบบมาเพื่อใช้เป็นตัวกลางในการจัดการผู้ใช้งานและเครื่องคอมพิวเตอร์แม่ข่ายที่ให้บริการเพื่อให้สามารถรองรับ load จากผู้ขอใช้บริการและกระจาย load ไปยัง Server ที่ให้บริการได้อย่างเหมาะสม ปัจจุบัน Load Balance มีทั้งแบบ hardware และ Software ซึ่งแต่ละผู้ผลิตก็จะมีขีดความสามารถที่แตกต่างกันออกไป แต่สำหรับสารนิพนธ์เล่มนี้นำเสนอ Load Balance แบบ Software Open Source เนื่องจากไม่มีค่าใช้จ่ายในเรื่องลิขสิทธิ์การใช้งานและมีขีดความสามารถที่ตอบสนองความต้องการในการให้บริการของ Web server ในปัจจุบัน

2. ทฤษฎีที่เกี่ยวข้อง

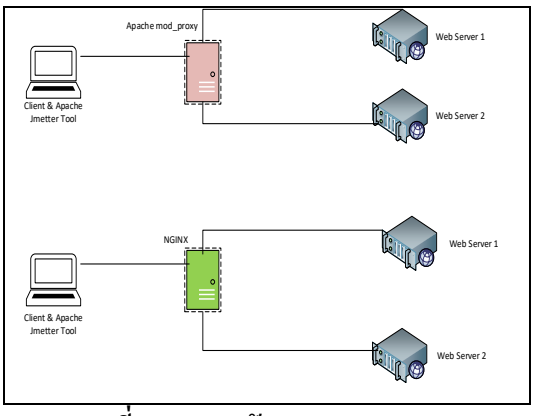
Proxy หรือ proxy server คือ อุปกรณ์หรือ server ที่คั่นกลางระหว่างผู้ใช้งาน กับเครื่องคอมพิวเตอร์ server ที่ให้บริการ ซึ่งหากไม่มี proxy server เครื่อง server ที่ทำหน้าเป็นผู้ให้บริการจะต้องติดต่อกับผู้ใช้งานโดยตรง และมีผลเสียมากกว่าผลดี load balance เป็น reverse proxy หลักการทำงานของ load balance เป็นการนำเอา server หลาย ๆ เครื่องมาทำงานร่วมกัน

โดยใช้วิธีการกระจาย load ไปแต่ละเครื่อง เพื่อให้สามารถรับภาระงานที่เข้ามาจาก user จำนวนมาก ๆได้[2] Apache mod_proxy เป็นโมดูลเสริมสำหรับ Apache HTTP Server โมดูลนี้ใช้ทำหน้าที่เป็น reverse proxy หรือแคช โดยทำงานร่วมกับ Apache web server แต่ก็สามารถนำมาใช้งานเป็น load balance สำหรับ Web server [3]อื่น ๆได้ NGINX เป็นซอฟต์แวร์โอเพนซอร์ส แบบ reverse proxy มีความสามารถในการ load balance โดยสามารถ load ได้ทั้งข้อมูล web server multimedia server หรือ streaming server และอื่น ๆ NGINX เป็น load balancer เซิร์ฟเวอร์ที่ออกแบบมาเพื่อให้มีประสิทธิภาพและความเสถียรภาพในการทำงานสูงสุด โดนสามารถ cache สำหรับอีเมล (IMAP, POP3 และ SMTP) คุณสมบัติในการ load balancer จะมีความสามารถ load ได้ทั้ง HTTP, TCP และ UDP [4]

Load Balance Method (Algorithm) เป็นตรรกะ การคิดหรือการแจกจ่ายภาระงานให้กับเครื่องที่ให้บริการที่อยู่ด้านหลัง load balance server ซึ่ง load balance แต่ละค่ายจะมี method หรือแนวทางการกระจายภาระงานที่แตกต่างกันออกไป การวัดขีดความสามารถ จะวัด throughput ซึ่งเป็นจำนวน transaction ต่อ request ที่ถูกสร้างขึ้นหรือทำงานได้ในช่วงเวลาการทดสอบหนึ่ง ๆ สำหรับบอกว่า ระบบ load balancer มีความสามารถในการจัดการงานจำนวนเท่าไรในแต่ละหนึ่งหน่วยเวลา

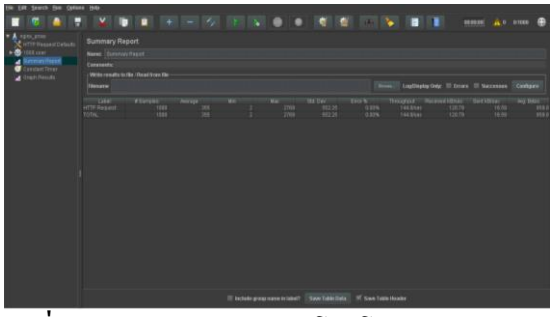
3. การดำเนินงาน

3.1การดำเนินการของงานวิจัยนี้จะเป็นการทดสอบเพื่อวัดขีดความสามารถของ Software load balance 2 ตัว คือ Apache mod_proxy และ NGINX ซึ่งการทดสอบจะทดสอบโดยเน้นไปที่ HTTP/1.1 เป็นหลัก แต่ก็จะทดสอบ HTTP/2 ด้วย การทดสอบทำโดยการวิธีวัดประสิทธิภาพจาก Throughput ที่สามารถรองรับได้ โดยมีการตั้งค่าพื้นเริ่มต้นที่เท่ากัน กำหนดให้ load balancer มีทรัพยากร ที่เท่ากันดังนี้คือ processor 2 core, memory 1GB และ hard disk 30GB ระบบปฏิบัติการ CentOS และกำหนด method สำหรับการload โดย Apache mod_proxy เลือกวิธีการ load แบบ byrequest, bytraffic และ bybusyness มาทำการทดสอบ ในส่วนของ NGINX เลือก แบบ round robin, least connection, IP Hash มาทดสอบ จากนั้นจะออกแบบรูปแบบการทดสอบเพื่อใช้ในการทดสอบ โดย web server ผู้ทดสอบใช้เลือกใช้ระบบปฏิบัติการ Microsoft Windows Server 2016 standard edition และติดตั้ง Internet Information Service (IIS) version 10 เพื่อทำหน้าที่เป็น web server พร้อมทั้งติดตั้ง web page ที่พัฒนาขึ้นมาเอง ด้วยภาษา HTML การวาง load balancer ในการทดสอบ จะวางไว้หน้า web server โดยมีการปรับเปลี่ยนค่า config ตาม method ที่ได้เลือกไว้แสดงดังภาพที่ 3.1



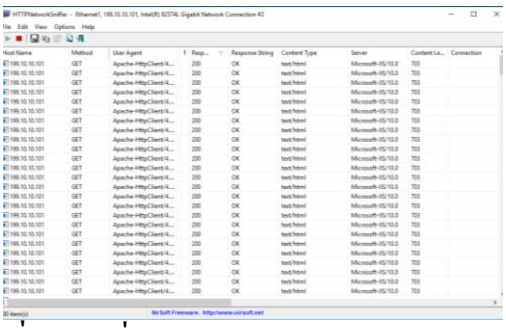
ภาพที่ 3.1 สถาปัตยกรรมของระบบ

3.2 การทดสอบจะยิง load และเก็บผลด้านประสิทธิภาพโดยใช้เครื่องมือ apache JMetter โดยสร้าง load 1000 user ส่ง request ไปยัง load balance เพื่อดูว่าสามารถรับ throughput ได้เท่าไหร่ ในแต่ละ method โดยการทดสอบจะแยกสภาพแวดล้อมออกจากกันเพื่อให้ได้ผลที่ดีที่สุด ทั้ง 2 สภาพแวดล้อมมีคุณสมบัติพื้นฐานที่หมือนกันยกเว้น load balancer ที่แตกต่างกัน ที่ HTTP/1.1 จะทดสอบ HTTP Method GET และ POST ไล่ลำดับไปตามแต่ละ method การทดสอบจะเน้นที่ HTTP/1.1 เป็นหลัก ในส่วนของ HTTP/2 การทดสอบต้องทำให้ load balancer ทั้ง 2 ค่าย ทำหน้าที่เป็น HTTP/2 web server ด้วย สำหรับการเก็บผลจะเก็บโดยพิจารณาจาก throughput ที่อยู่บน software Apache JMetter ตามภาพที่ 3.2



ภาพที่ 3.2 การทดสอบระบบโดยโปรแกรม JMetter

ในการวัดประสิทธิผล หรือแต่ละเครื่อง backend มีภาระงานที่ได้รับเป็นจำนวนเท่าไหร่จะใช้เครื่องมือ HTTPNetworkSniffer ซึ่งเป็น Software สำหรับเก็บค่าจำนวนภาระงานที่ถูกจ่ายไปยัง backend server โดยดูจาก HTTP response code 200 ดังภาพที่ 3.3



ภาพที่ 3.3 เครื่องมือตรวจสอบการกระจายภาระงาน

การเก็บผลด้านการใช้งาน resource ด้วยโดยจะดูจาก system monitoring ซึ่งเป็นเครื่องมือที่มาพร้อมกับ CentOS

4. ผลทดการสอบ

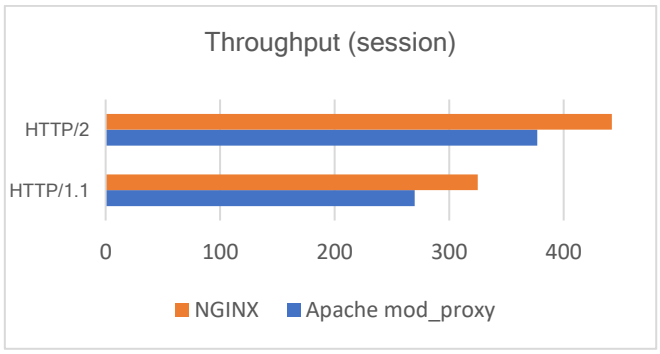
ผลการทดสอบการวัดประสิทธิภาพการทำงานของ Software Load Balance ระหว่าง Apache mod_proxy และ NGINX สำหรับ Web server สามารถแยกผลการทดสอบตามรูปแบบต่าง ๆ ดังต่อไปนี้

4.1 ผลทดสอบประสิทธิภาพด้านความสามารถในการรับจำนวนผู้ใช้งาน (Throughput) ของ Apache mod_proxy ของ HTTP/1.1 Apache mod_proxy method GET จะมี Throughput ที่สูงกว่า method POST ในทุก ๆวิธีการ load และจะสูงที่สุดในแบบ ซึ่งมีความเป็นไปได้ว่า เนื่องจาก method GET มีการตอบสนองข้อมูลกลับมาที่มากกว่า แต่ Method POST มีการส่งค่าหรือข้อมูลบางอย่างไปยัง web site จึงทำให้การตอบสนองข้อมูลได้น้อยกว่า GET ด้านการใช้งาน CPU มีการใช้งาน CPU ที่แตกต่างกันโดยการ load แบบ byrequests method GET ใช้ CPU สูงกว่า method POST โดยต่างกัน 8% การ load แบบ bybusyness method POST ใช้ CPU สูงกว่า method GET โดยต่างกันที่ 8.6% แต่ในการ load แบบ bytraffic มีการใช้งาน CPU ที่ใกล้เคียงกันคือมีความแตกต่างกันที่ 0.8% ด้านการใช้งาน memory Apache mod_proxy มีการใช้งาน memory ที่ใกล้เคียงกันในทุกรูปแบบ การตอบสนองต่อผู้ใช้งาน method GET Response เร็วกว่า POST 2 รูปแบบคือ byrequest และ bybusyness โดยการ load แบบ byrequest มีความแตกต่างกันระหว่าง method GET กับ POST ที่ 1.2ms bybusiness มีความแตกต่างกันระหว่าง method GET กับ POST ที่ 0.2 ms แต่ bytraffic Response Time เท่ากันคือ 4.5ms

4.2 ผลทดสอบประสิทธิภาพด้านความสามารถในการรับจำนวนผู้ใช้งาน (Throughput) ของ NGINX บน ของ HTTP/1.1 NGINX method GET จะมี Throughput ที่สูงกว่า method POST 2 วิธีการ load คือ Round Robin และ IP Hash โดยการload แบบ Round Robin method GET จะแตกต่างจาก POST 54 session และ IP Hash method GET จะแตกต่างจาก POST 16 session ในส่วนของการ load แบบ Lest Connection method POST มีค่า

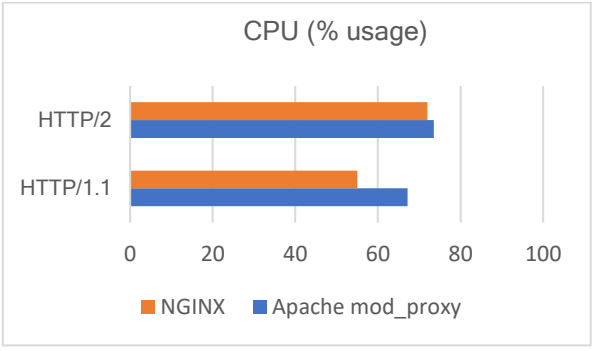
Throughput ที่สูงกว่า GET โดยแตกต่างกันที่ 18 session ด้านการใช้งาน CPU มีการใช้งาน CPU ที่แตกต่างกันแต่ไม่มากโดยสูงสุดอยู่ที่ การ load แบบ Round Robin method POST ใช้ CPU สูงกว่า method GET โดยต่างกัน 2.4% ด้านการใช้งาน memory มีการใช้งาน memory ที่ใกล้เคียงกันทุปรูปแบบการ load การตอบสนองต่อผู้ใช้งานพบว่า Apache mod_proxy method GET Response เร็วกว่า POST 2รูปแบบ Round Robin และ IP Hash

4.3 ผลทดสอบประสิทธิภาพด้านความสามารถในการรับจำนวนผู้ใช้งาน (Throughput) โดยการเปรียบเทียบระหว่าง HTTP/1.1 และ HTTP/2 NGINX มีประสิทธิภาพในการรับ throughput ได้สูงกว่า Apache mod_proxy โดยที่ HTTP/1.1 NGINX มีความแตกต่างกับ Apache mod_proxy 55 session ที่ HTTP/2 NGINX มีความแตกต่างกับ Apache mod_proxy 65 session ดังภาพที่ 4.1



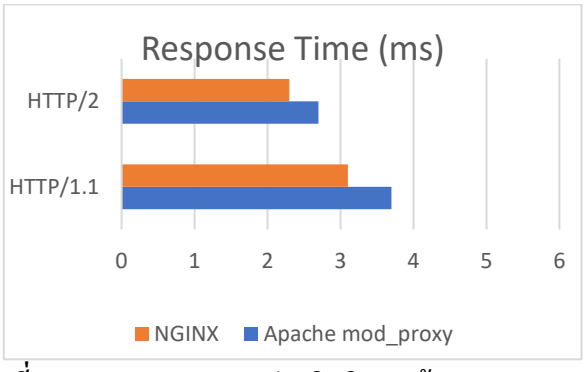
ภาพที่ 4.1 ผลการทดสอบประสิทธิภาพด้านรับจำนวนผู้ใช้งาน (Throughput)

ด้านการใช้งาน CPU พบว่า NGINX มีการใช้ CPU resource ที่น้อยกว่า Apache mod_proxy โดยที่ HTTP/1.1 NGINX มีความแตกต่างกับ Apache mod_proxy 12% ที่ HTTP/2 NGINX มีความแตกต่างกับ Apache mod_proxy 1.5% ดังภาพที่ 4.2



ภาพที่ 4.2 ผลการทดสอบประสิทธิภาพด้านการใช้งาน Processor

ด้านการใช้งาน Memory Apache mod_proxy และ NGINX มีการใช้งาน memory ที่ใกล้เคียงกัน การตอบสนองต่อผู้ใช้งานNGINX มีการช่วงเวลา Response Time (Latency)ที่น้อยกว่า Apache mod_proxy โดยที่ HTTP/1.1 NGINX มีความแตกต่างกับ Apache mod_proxy ที่ 0.6ms ที่ HTTP/2 NGINX มีความแตกต่างกับ Apache mod_proxy ที่ 0.4ms ดังภาพที่ 4.3



ภาพที่ 4.3 ผลการทดสอบประสิทธิภาพด้านการตอบสนอง

4.4 ผลการทดสอบประสิทธิผลการกระจายภาระงานของ Software Load balance NGINX มีการกระจายภาระงานไปยัง backend server ได้ค่อนข้างสมดุลมากกว่า Apache mod_proxy โดยพบว่า ที่ภาระงานจำนวน 1000 session มีความแตกต่างระหว่าง backend server 1 และ backend server 2 ไม่มาก หากเทียบกับ Apache mod_proxy ซึ่งจะเห็นชัดเจนในรูปแบบการ load แบบ bytraffic และ แบบ bybusiness จำนวนภาระงานระหว่าง Backend server 2 เครื่องมีความแตกต่างกันมาก

4.5การเปรียบเทียบ Method (Algorithm) Software Apache mod_proxy และ NGINX ต่างมีความสามารถ ในการใช้งาน Load balance โดยสามารถเลือก method ที่ใช้ในการ load ได้ตามตารางที่ 4.1

Software	Apache mod_proxy	NGINX
Method (Algorithms)	byrequests	Round Robin
	bytraffic	Least Connection
	bybusyness	Least Time
	heartbeat	Hash
		IP Hash
		Random

ตารางที่ 4.1 ผลการเปรียบเทียบ method

4.6 ผลเปรียบเทียบประสิทธิภาพโดยรวม

คุณลักษณะ	Apache mod_proxy	NGINX
ความสามารถในการ load	รองรับ session ที่เข้ามาใช้งานได้น้อยกว่า NGINX	รองรับ session ที่เข้ามาใช้งานได้ค่อนข้างสูงกว่าประมาณ 15.6%
การใช้งาน processor	ใช้หน่วยประมวลผลสูง	ใช้หน่วยประมวลผลน้อย
การใช้งาน memory	ใช้ memory ค่อนข้างสูง	ใช้ memory ค่อนข้างน้อย
การเพิ่ม module อื่นๆ เพื่อให้ทำงานได้มากกว่าการเป็น load balance ในserver เดียวกัน	สามารถทำได้เนื่องจาก มี module (mod_xxx)ต่างๆที่สามารถติดตั้งใช้งานร่วมด้วยได้	ต้องใช้ software อื่นร่วมด้วยเนื่องจาก ถูกออกแบบมาให้ทำหน้าที่ load balance เพียงอย่างเดียว

ตารางที่ 4.2 ผลการเปรียบเทียบประสิทธิภาพ

5. ข้อเสนอแนะ

5.1 ในการใช้งาน load balance แม้ว่า load balance จะมี method ในการ load ที่หลากหลายแต่หากเลือกใช้ไม่เหมาะสม อาจจะทำให้การกระจาย load ไปยังเครื่อง backend ไม่ดีพอ

5.2 การออกแบบระบบที่ดี ควรจะมี backend อย่างน้อย 2 ชุดเพื่อป้องกันและแต่ละชุดควรจะออกแบบให้รองรับ load ได้ 50:50 เพื่อป้องกันปัญหา หาก server เสียหายไป 1 ชุด จะยังมีอีก 1 ที่สามารถทำงานทดแทนกันได้

5.3 การออกแบบระบบควรจะมี load balance ในระบบ 2 ชุดเพื่อป้องกันปัญหา single point of failure เพราะ load balance เป็น reverse proxy จะถูกจัดวางในตำแหน่ง หน้าสุด ซึ่งหาก service ของ load balance ทำงานไม่ได้ แต่ service ของ web server ไม่ได้เสียหาย ผู้ใช้งานจะเข้าใช้งานไม่ได้ ควรจะออกแบบให้ load balance เป็นแบบ HA ซึ่งอาจจะเป็นในรูปแบบใด รูปแบบ หนึ่งคือ active-active หรือ active standby

5.4 หาก backend เป็น service อื่นๆ ที่ไม่ใช่ apache หรือ NGINX การเลือกใช้งานถ้าผู้ใช้ ต้องการจะ implement load balance พร้อม web application firewall Apache mod_proxy จะเหมาะสมกว่า เพราะมี mod_security

5.5ในการออกแบบระบบปัจจุบัน server ส่วนใหญ่จะอยู่ในรูปแบบของ virtualization environment สำหรับ load balance สามารถทำงานได้กับ virtualization environment แต่ควรจะแยก port หรือ ทำ Qos network port เพราะ load balance จะต้องรับ network traffic ที่สูงกว่า server โดยทั่วไป

6. อ้างอิง

[1] Google chome, “January 2020 Web server Survey” 2020. [Online]. Available:https: //https://en.wikipedia.org/wiki/N%2B1_redundancy [Accessed: 5-Jan-2020].

[2] Adam Pi´orkowski1, Aleksander Kempny2, Adrian Hajduk1, and Jacek Strzelczyk1 : ″Load Balancing for Heterogeneous Web servers″ University of Muenster, Muenster, Germany

[3] Google chome, “January 2020 Web server Survey” 2020. [Online]. Available: https://httpd.apache.org/docs/2.4/mod/mod_proxy.html [Accessed: 12-Jan-2020].

[4] Google chome, “January 2020 Web server Survey” 2020. [Online].Available: https://www.nginx.com/blog/whats-difference-nginx-foss-nginx-plus [Accessed: 12-Jan-2020