การพิจารณาประสิทธิภาพการทำงานของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

พชรวร บุญชู และ ผศ.มยุรี เลิศเวชกุล

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง กรุงเทพฯ
Emails: s8061054@kmitl.ac.th, klmayure@kmitl.ac.th

บทคัดย่อ

เว็บเซอร์วิสเป็นเทคโนโลยีที่กำลังได้รับความนิยมสำหรับการพัฒนาเว็บแอพพลิเคชันเพื่อให้บริการแลกเปลี่ยนข้อมูลผ่านระบบเครือข่ายอินเทอร์เน็ต แต่เนื่องด้วยในปัจจุบันยังคงมีการใช้งานอินเทอร์เน็ตบนระบบเครือข่ายที่มีความเร็วจำกัดอยู่เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งการใช้ GPRS/EDGE เพื่อเชื่อมต่ออินเทอร์เน็ตผ่านโทรศัพท์มือถือที่มีคุณภาพการเชื่อมต่อที่ต่ำแต่มีค่าใช้จ่ายที่ค่อนข้างสูง ดังนั้นการพิจารณารูปแบบของการให้บริการเว็บเซอร์วิสที่เหมาะสมจึงเป็นสิ่งสำคัญ ผู้วิจัยจึงได้ศึกษาถึงข้อดีของการนำทฤษฎีสถาปัตยกรรม REST มาใช้ในการพัฒนาเว็บเซอร์วิส และทำการทดสอบเปรียบเทียบประสิทธิภาพของการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และสถาปัตยกรรมแบบ SOAP โดยผลการทดสอบแสดงให้เห็นว่าการให้บริการข้อมูลด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นสามารถช่วยลดระยะเวลาที่ใช้ในการร้องขอบริการจากเว็บเซอร์วิสและเพิ่มประสิทธิภาพในการให้บริการของแม่ข่ายเว็บเซอร์วิสได้ถึงประมาณ 30 เปอร์เซนต์ อีกทั้งยังเป็นการลดปริมาณข้อมูลที่ต้องรับส่งระหว่างเครือข่ายได้มากถึงประมาณ 90 เปอร์เซนต์ เมื่อเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

คำสำคัญ-- REST; SOAP; เว็บเซอร์วิส; ประสิทธิภาพ

1. บทนำ

แนวโน้มสำหรับการให้บริการเว็บ (World Wide Web) [1] ในปัจจุบันได้มีการให้บริการเนื้อหาของเว็บที่เป็นแบบไดนามิคเพิ่มมากขึ้นอย่างต่อเนื่อง และเทคโนโลยีที่เป็นที่นิยมสำหรับการพัฒนาเว็บแอพพลิเคชันในลักษณะดังกล่าวเรียกว่า เว็บเซอร์วิส (Web service) โดยที่เว็บเซอร์วิสนั้นคือเว็บแอพพลิเคชันที่เป็นแม่ข่ายในการให้บริการเนื้อหาที่ผู้ใช้งานสามารถเข้าถึงได้โดยโปรโตคอลสื่อสารอย่าง เช่น Hypertext transfer protocol (HTTP) [2] เป็นต้น

เนื่องจากในปัจจุบันยังคงมีการใช้งานอินเทอร์เน็ตที่มีข้อจำกัดในเรื่องความเร็วในการใช้บริการและปริมาณการรับส่งข้อมูลที่จำกัดอยู่เป็นจำนวนมาก โดยเฉพาะอย่างยิ่งการใช้ GPRS หรือ EDGE เพื่อเชื่อมต่ออินเทอร์เน็ตผ่านโทรศัพท์มือถือที่มีคุณภาพในการเชื่อมต่อที่ต่ำแต่มีค่าใช้จ่ายที่ค่อนข้างสูง ด้วยเหตุนี้เองทำให้เกิดการใช้งานเว็บเซอร์วิสจึงถูกใช้งานอยู่บนการเชื่อมต่อที่มีข้อจำกัด ทำให้การร้องขอใช้บริการเว็บเซอร์วิสอาจจะใช้เวลานานเกินไป หรือ อาจจะไม่ได้รับการให้บริการจากแม่ข่ายเว็บเซอร์วิสเลยในบางครั้ง

ส่วนสำคัญในงานวิจัยฉบับนี้คือการนำทฤษฎีสถาปัตยกรรม REST มาใช้ในการพัฒนาเว็บเซอร์วิสเพื่อเป็นแนวทางในแก้ไขปัญหาการใช้งานเว็บเซอร์วิสบนการเชื่อมต่อที่มีคุณภาพต่ำ และทำการทดสอบประสิทธิภาพการทำงานของ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ดังกล่าวเมื่อเปรียบเทียบกับการร้องขอบริการผ่าน เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

2. เว็บเซอร์วิส

เว็บเซอร์วิสเป็นซอฟต์แวร์ที่ถูกพัฒนาขึ้นเพื่อให้บริการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ผ่านระบบเครือข่ายอินเทอร์เน็ตด้วยการร้องขอบริการเว็บเซอร์วิสในรูปแบบของ Simple Object Access Protocol (SOAP) [3] หรือ ส่งการร้องขอบริการเว็บเซอร์วิสตามทฤษฎีของสถาปัตยกรรม Representational State Transfer (REST) [1] เป็นต้น ทั้งนี้จะอาศัยรูปแบบภาษามาตรฐานที่เรียกว่า Extensible Markup Language (XML) ในการส่งเนื้อหาของการร้องขอบริการนั้น ทำให้เครื่องคอมพิวเตอร์ที่เรียกใช้งานเว็บเซอร์วิสสามารถสื่อสารกันได้ถึงแม้ว่าจะมีแพลตฟอร์มที่แตกต่าง หรือถูกพัฒนาด้วยภาษาโปรแกรมที่ต่างกันก็ตาม

2.1 สถาปัตยกรรม SOAP

Simple Object Access Protocol (SOAP) คือโปรโตคอล หรือวิธีการมาตรฐานที่ใช้ในการสื่อสารกันระหว่างเว็บเซอร์วิสโดยอาศัยการส่งข้อมูลผ่านโปรโตคอลสื่อสารอย่างเช่น HTTP หรือ SMTP เป็นต้น แต่โดยทั่วไปแล้วจะใช้การส่งบน HTTP เนื่องจากเป็นโปรโตคอลที่แพร่หลายและมีการใช้งานอยู่แล้วบนอินเทอร์เน็ต โดยเมื่อพัฒนาโปรแกรมด้วยรูปแบบของ SOAP ในภาษาโปรแกรมที่ใช้ และใช้งานร่วมกับ Web Services Description Language (WSDL) [7] จากนั้น SOAP ก็จะสร้างข้อความ SOAP (SOAP Message) [6] เพื่อติดต่อกับแอพพลิเคชันปลายทางให้โดยอัตโนมัติ

ข้อความ SOAP (SOAP Envelope) ที่บรรจุข้อมูลไว้ภายในแบ่งออกได้เป็น 2 ส่วน คือ ส่วนหัว SOAP (SOAP Header) อันประกอบด้วยคำอธิบาย หรือ รายละเอียดต่างๆ และส่วนที่สอง คือ เนื้อหา SOAP (SOAP Body) ซึ่งประกอบด้วยเนื้อหาของการร้องขอหรือการตอบกลับของเว็บเซอร์วิสในรูปแบบของภาษามาตรฐาน XML

2.2 สถาปัตยกรรม REST

Representational State Transfer (REST) เป็นรูปแบบทฤษฎีของสถาปัตยกรรมทางซอฟต์แวร์สำหรับการนำเสนอสื่อต่างๆ เช่น เอกสาร เอชทีเอ็มแอล โดยที่ REST เองนั้นไม่ได้ถูกจำกัดให้ใช้งานกับโปรโตคอลใดเป็นพิเศษ แต่ส่วนใหญ่ก็จะใช้งานร่วมกับ HTTP ที่มีลักษณะการทำงานแบบการร้องขอและตอบรับ (Request-Response) และตัวเนื้อหาของสื่อที่จะส่งไปนั้นสามารถเป็นข้อมูลชนิดใดก็ได้ตามที่ต้องการ เช่น HTML, Plain Text, XML หรือ JSON เป็นต้น

เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST [4] (ในบางครั้งเรียกว่า RESTful) จะเน้นไปที่ผลลัพธ์ของการให้บริการ โดยจะต้องสามารถกำหนดที่อยู่ของทรัพยากรในระบบให้สามารถเข้าถึงได้ด้วยการกำหนดคีย์หลักที่ไม่ซ้ำกันให้กับทรัพยากรทั้งหมด แล้วระบุที่อยู่ผ่านทาง URL [5] เท่านั้น ดังนั้นทุกครั้งที่ต้องการร้องขอใช้บริการหรืออ้างอิงผลลัพธ์ของการให้บริการ ก็เพียงแค่ระบุ URL ร่วมกับกระบวนการทำงานของ HTTP (HTTP Method) เท่านั้น

2.3 ทฤษฎีสถาปัตยกรรม REST

พื้นฐานของการออกแบบสถาปัตยกรรม REST นั้นเน้นที่การใช้งานกระบวนการทำงานของ HTTP (POST, GET, PUT และ DELETE) ให้ถูกต้องและเหมาะสม, การไม่จัดเก็บสถานะการทำงาน (Stateless), การใช้รูปแบบของ URL ที่มีลักษณะคล้ายกับโครงสร้างของไดเรคทอรี่ และ การนำสามารถเลือกใช้ข้อมูลมาตรฐานชนิดในก็ได้ในการแสดงเนื้อหาของการร้องขอบริการและการตอบกลับ ซึ่งมีรายละเอียดดังต่อไปนี้

2.3.1 การกำหนดรูปแบบของ URL ที่คล้ายกับโครงสร้างไดเรคทอรี่

การใช้ URL เป็นตัวชี้ไปที่วัตถุ ทรัพยากร หรือ บริการต่างๆ โดยสามารถกำหนดกระบวนการของ HTTP ไปพร้อมกับการร้องขอเพื่อเป็นการแสดงว่าต้องการให้แม่ข่ายเว็บเซอร์วิสดำเนินการอย่างไรกับทรัพยากรดังกล่าว

2.3.2 การไม่จัดเก็บสถานะการทำงาน (Stateless)

เครื่องแม่ข่ายจะไม่เก็บข้อมูลเซสชันของการใช้บริการของผู้ใช้ไว้ โดยเครื่องแม่ข่ายจะจัดการเพียงสถานะของทรัพยากรที่ดูแลอยู่เท่านั้น ซึ่งในกรณีที่แอพพลิเคชันมีความต้องการที่จะเก็บข้อมูลเซสชันของการใช้บริการของผู้ใช้นั้น สามารถทำได้ด้วยการเก็บไว้ที่เครื่องลูกข่ายเอง และส่งมาที่เครื่องแม่ข่ายพร้อมกับการร้องขอบริการเมื่อต้องการใช้งาน ซึ่งลักษณะการทำงานเช่นนี้จะช่วยทำให้ระบบสามารถขยายการให้บริการในอนาคตได้ง่าย (Scalable)

2.3.3 กำหนดวิธีการใช้งานกระบวนการทำงานของ HTTP อย่างชัดเจน

การใช้งานกระบวนการต่างๆของ HTTP นั้นจะต้องเป็นไปตามรายละเอียดที่แสดงในตาราง 1.

ตาราง 1. มาตรฐานการใช้งานกระบวนการของ HTTP

กระบวนการ	รายละเอียด
POST	เป็นกระบวนการที่ใช้สั่งให้เครื่องแม่ข่ายทำการเก็บข้อมูล ซึ่งสามารถเลือกได้ว่าจะส่งหรือไม่ส่งข้อมูลไปพร้อมกับการร้องขอก็ได้
GET	เป็นกระบวนการทำงานเพื่อเรียกดูข้อมูลที่ต้องการจากเครื่องแม่ข่าย
PUT	เป็นกระบวนการที่ใช้สั่งให้เครื่องแม่ข่ายทำการสร้างหรือแก้ไขข้อมูลด้วยข้อมูลที่ส่งมาพร้อมกับการร้องขอ
DELETE	เป็นกระบวนการที่ใช้สำหรับสั่งให้เครื่องแม่ข่ายทำการลบข้อมูล

โดยกระบวนการ GET, PUT และ DETETE จะต้องอ้างอิงตำแหน่งของทรัพยากรด้วยค่าของคีย์หลัก

2.3.4 สามารถใช้กับรูปแบบของข้อมูลมาตรฐานชนิดใดก็ได้

ส่วนแสดงเนื้อหาของการร้องขอและการตอบกลับนั้นสามารถเลือกใช้ข้อมูลชนิดใดก็ได้ที่สามารถอ้างอิง MIME-type และ Content-Type ได้ เช่น HTML, Plain Text หรือ JSON เป็นต้น

3. การทดสอบประสิทธิภาพของการให้บริการเว็บเซอร์วิสด้วยรูปแบบของ SOAP และ REST

สำหรับในข้อหัวนี้จะกล่าวถึงการทดสอบเปรียบเทียบระหว่างประสิทธิภาพของการให้บริการเว็บเซอร์วิสด้วยรูปแบบตามมาตรฐานของ SOAP และรูปแบบตามแนวทางทฤษฎีของ REST ที่ได้กล่าวไว้ในหัวข้อที่ 2. โดยจะแสดงให้เห็นถึงวิธีการทดสอบและผลการทดสอบที่ได้จากการทดสอบ

3.1 ระบบที่ใช้ในการทดสอบ

ผู้วิจัยได้ทำการทดสอบประสิทธิภาพของเว็บเซอร์วิสที่ใช้สถาปัยกรรม REST โดยเปรียบเทียบกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP โดยทดสอบให้เครื่องลูกข่ายเชื่อมต่อกับเครื่องแม่ข่ายที่ให้บริการเว็บเซอร์วิสตามสถาปัตยกรรมทั้งสองแบบผ่านเราท์เตอร์

โดยเครื่องแม่ข่ายใช้คอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเป็น Intel® Core™2 Duo E6300 1.86 GHz. หน่วยความจำ 2GB ทำงานบนระบบปฏิบัติการ Windows XP Professional Service Pack 3 และใช้ซอฟต์แวร์ Apache Web Server เวอร์ชัน 2.2.8 ที่รองรับภาษา PHP เวอร์ชัน 5.2.6 เป็นซอฟต์แวร์ระบบแม่ข่ายเว็บ โดยใช้ระบบจัดการฐานข้อมูล MySQL เวอร์ชัน 5.0.51b

เครื่องลูกข่ายใช้คอมพิวเตอร์ที่มีหน่วยประมวลผลกลางเป็น Intel® Core™2 Duo T9300 2.50 GHz. หน่วยความจำ 2GB ทำงานบนระบบปฏิบัติการ Windows XP Professional Service Pack 3 และใช้ซอฟต์แวร์ Apache JMeter [8] เวอร์ชัน 2.3.4 ในการทดสอบและบันทึกผลการทดสอบ

3.2 แม่ข่ายเว็บเซอร์วิส

การทำงานของเว็บเซอร์วิสแบ่งออกเป็น 4 บริการ คือ CREATE, RETRIEVE, UPDATE และ DELETE ในที่นี้ใช้การจำลองการสร้างและเก็บข้อมูลสมาชิกซึ่งประกอบไปด้วย หมายเลขโทรศัพท์, เลขประจำตัวประชาชน, ชื่อ, นามสกุล, ชื่อประเทศ, รหัสไปรษณีย์, อีเมล์ และวันที่ในการทำรายการ โดยการใช้หมายเลขโทรศัพท์เป็นคีย์หลัก (Primary Key :PK) ของตารางดังกล่าว

บริการ CREATE ใช้ในการสร้างข้อมูลสมาชิก บริการ UPDATE ใช้แก้ไขข้อมูลสมาชิกที่มีอยู่แล้ว บริการ DELETE ใช้ในการลบข้อมูลสมาชิก โดยที่บริการทั้ง 3 นี้จะส่งผลลัพธ์ของการสร้าง การแก้ไข และการลบกลับไปยังเครื่องลูกข่าย ส่วนบริการ RETRIEVE เป็นบริการในการเรียกดูข้อมูลสมาชิก ซึ่งในกรณีที่ไม่พบข้อมูลสมาชิกจะส่งรหัสที่บ่งบอกว่าไม่พบข้อมูลสมาชิกกลับไปยังเครื่องลูกข่ายเป็นผลลัพธ์

เว็บเซอร์วิสที่ใช้ทำการทดสอบนั้นพัฒนาด้วยภาษา PHP เวอร์ชัน 5 โดยใช้ฟังก์ชั่นมาตรฐานของภาษา PHP เองทั้งหมด โดยได้ทำการพัฒนาเว็บเซอร์วิสออกเป็นสองชุด ซึ่งเป็นไปตามสถาปัตยกรรม SOAP และสถาปัตยกรรม REST สำหรับเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP จะเรียกใช้งานคลาส SoapServer พร้อมอ้างอิงไฟล์ WSDL ที่ระบุถึงบริการจากบริการ 4 ชนิด (CREATE, RETRIEVE, UPDATE และ DELETE) ส่วนเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นใช้การร้องขอตามมาตรฐานของ HTTP ในภาษา PHP และใช้คลาส SimpleXMLElement ในการอ่านเนื้อหาของการร้องขอ โดยมีบริการ 4 บริการเช่นเดียวกันกับเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP

3.3 แม่ข่ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่ใช้ทดสอบนั้น ผู้วิจัยได้ทำการพัฒนาขึ้นด้วยการอ้างอิงตามพื้นฐาน 4 ข้อของทฤษฎีสถาปัตยกรรม REST ดังที่ได้กล่าวมาแล้วข้างต้น โดยการจับคู่บริการของเว็บเซอร์วิสตามสถาปัตยกรรม REST กับกระบวนการทำงานของ HTTP เป็นไปดังที่แสดงในตาราง 2.

ตาราง 2. การจับคู่บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST กับกระบวนการ HTTP

บริการ	กระบวนการ HTTP	รายละเอียดการทำงาน
CREATE	POST	สร้างข้อมูลสมาชิกในระบบ
RETRIEVE	GET	เรียกดูข้อมูลสมาชิกในระบบ
UPDATE	PUT	แก้ไขข้อมูลสมาชิกในระบบ
DELETE	DELETE	ลบข้อมูลสมาชิกในระบบ

โดยทั้งนี้การออกแบบเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ในการทดสอบนั้น สถานะการทำงาน ค่าข้อมูลควบคุมการทำงานเซสชันของผู้ใช้งาน และพารามิเตอร์ทุกตัวจะถูกจัดการและส่งมาจากเครื่องลูกข่ายเท่านั้นโดยการอ้างอิงทรัพยากรและบริการของเว็บเซอร์วิสจะเป็นไปตามโครงสร้างของ URL ที่กำหนดดังในรูปที่ 1. ซึ่งมีรูปแบบที่คล้ายกับโครงสร้างไดเรคทอรี่ ซึ่งรูปที่ 2. เป็นการแสดงตัวอย่างการเข้าถึงเครื่องแม่ข่ายที่ชื่อ “TESTSERV” โดยเรียกบริการชื่อ “FREECONTENT” ที่ทำงานที่พอร์ตหมายเลข “88” เพื่ออ้างอิงถึงข้อมูลของหมายเลขโทรศัพท์ “66810987654”

```
HTTP : // IP : PORT / SERVICE / RESOURCE
```

รูปที่ 1. การกำหนดโครงสร้างของ URL ของเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST

```
HTTP : // TESTSERV : 88 / FREECONTENT / 66810987654
```

รูปที่ 2. ตัวอย่างการกำหนด URL ตามโครงสร้าง URL ที่กำหนด

และสุดท้ายเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ที่พัฒนาขึ้นมาเพื่อทดสอบนั้น ผู้วิจัยได้เลือกใช้ภาษามาตรฐาน XML ในการแสดงส่วนเนื้อหาของการร้องขอและการตอบกลับ โดยอ้างอิงจากเนื้อหาของการร้องขอและการตอบกลับจาก เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP ให้ได้ค่าพารามิเตอร์ที่ครบถ้วนเช่นเดียวกัน เพื่อให้ได้ผลลัพธ์จากการร้องขอบริการของเว็บเซอร์วิสทั้งสองสถาปัตยกรรมที่เหมือนกัน ดังตัวอย่างของการร้องขอและการตอบกลับของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ในรูปที่ 3. และ 4. และตัวอย่างของการร้องขอและตอบกลับของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP ที่แสดงในรูป 5. และ 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<subscriber>
   <msisdn>66810987654</msisdn>
   <id>3100001000990</id>
   <title>mr</title>
   <firstname>REST</firstname>
   <lastname>TEST</lastname>
   <country> Thailand </country>
   <postcode>10220</postcode>
   <email>66810987654@rest.com</email>
   <active_date>2010/07/01 08:30:00</active_date>
</subscriber>
```

รูปที่ 3. ตัวอย่างเนื้อหาการร้องขอบริการ CREATE
ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

```
<?xml version="1.0" encoding="UTF-8"?>
<subscriber>
   <result>0</result>
</subscriber>
```

รูปที่ 4. ตัวอย่างเนื้อหาการตอบกลับบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="urn:xmethods-delayed-quotes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <ns1:create>
      <msisdn xsi:type="xsd:string">66810987654</msisdn>
      <id xsi:type="xsd:string">3100001000990</id>
      <title xsi:type="xsd:string">mr</title>
      <firstname xsi:type="xsd:string">SOAP</firstname>
      <lastname xsi:type="xsd:string">TEST</lastname>
      <country xsi:type="xsd:string">Thailand</country>
      <postcode xsi:type="xsd:string">10220</postcode>
      <email xsi:type="xsd:string">66810987654@soap.com</email>
      <active_date xsi:type="xsd:datetime">2010-07-
01T08:30:00</active_date>
    </ns1:create>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ 5. ตัวอย่างเนื้อหาการร้องขอบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ns1="urn:xmethods-delayed-quotes"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <ns1:createResponse>
      <result xsi:type="xsd:string">0</result>
    </ns1:createResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

รูปที่ 6. ตัวอย่างเนื้อหาการตอบกลับบริการ CREATE ผ่านเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

3.4 วิธีการทดสอบ

ผู้วิจัยได้ทำการทดสอบโดยใช้ซอฟต์แวร์ Apache JMeter เพื่อสร้างการร้องขอบริการไปยังแม่ข่ายเว็บเซอร์วิสทั้ง 2 สถาปัตยกรรม แล้วเก็บผลลัพธ์ที่แม่ข่ายเว็บเซอร์วิสตอบกลับมาไว้ในเครื่องลูกข่าย โดยทำการแยกทดสอบทีละสถาปัตยกรรม และทดสอบรอบละหนึ่งบริการเท่านั้น ด้วยการร้องขอบริการจำนวน 25,000 ครั้งต่อการทดสอบ 1 รอบ และทำการทดสอบซ้ำทั้งหมด 3 รอบสำหรับหนึ่งบริการของเว็บเซอร์วิสหนึ่งสถาปัตยกรรม

รูปแบบของการร้องขอทั้งหมดนั้นจะใช้การเชื่อมต่อ HTTP แบบทรานแซคชัน โดยไม่มีการเก็บแคชของ HTTP และมีส่วนหัวของการเชื่อมต่อ HTTP (HTTP header) ขนาด 160ไบต์ โดยประมาณใกล้เคียงกันทั้งสำหรับการร้องขอไปยังแม่ข่าย เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP และแม่ข่าย เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ซึ่งส่วนหัวนี้จะไม่ถูกนำมาใช้ในการวิเคราะห์ผลการทดสอบ โดยการทดสอบจะพิจารณาเฉพาะขนาดของเนื้อหาที่อยู่ภายในเท่านั้น

4. ผลการทดสอบ

4.1 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

ถ้าพิจารณาจากเรื่องขนาดของเนื้อหา (Content-Length) ที่ส่งการร้องขอบริการออกไปนั้นสามารถแบ่งได้เป็นสองกลุ่มคือกลุ่มของบริการ CREATE และบริการ UPDATE ซึ่งสองบริการนี้จะมีเนื้อหาที่ส่งไปขนาดประมาณ 950 ไบต์เนื่องจากต้องส่งข้อมูลสมาชิกไปด้วย เพื่อทำการสร้างหรือแก้ไข โดยที่การตอบกลับจากแม่ข่ายเว็บเซอร์วิสจะส่งผลลัพธ์ของการสร้างหรือแก้ไขกลับมาด้วยเนื้อหาขนาดประมาณ 515 ไบต์ ส่วนบริการ RETRIEVE และบริการ DELETE จะเป็นบริการที่มีการส่งการร้องขอที่มีเนื้อหาในขนาดที่เล็กกว่าเพราะมีการส่งเพียงแค่คีย์หลักเท่านั้น ส่วนการตอบกลับจากแม่ข่ายเว็บเซอร์วิสนั้นถ้าเป็นบริการ DELETE จะส่งผลลัพธ์ของการลบกลับมาด้วยเนื้อหาขนาดประมาณ 515 ไบต์ และบริการ RETRIEVE จะส่งข้อมูลสมาชิกกลับมาเป็นผลลัพธ์จึงทำให้มีขนาดเนื้อหาตอบกลับที่ค่อนข้างใหญ่ประมาณ 1,460 ไบต์

เวลาที่ใช้ในการทำรายการ (Latency) แต่ละรายการเฉลี่ยโดยรวมจากการทดสอบทั้งสามรอบนั้นพบว่า เวลาในการตอบสนองบริการแต่ละครั้งนั้นไม่แตกต่างกันมากนัก โดยจะใช้เวลาทั้งสิ้น 21-24 มิลลิวินาที ส่วนปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาที (Transactions per second) นั้นอยู่ที่ประมาณ 40-45 ทรานแซคชันต่อวินาที ดังจะแสดงค่าเฉลี่ยจากการทดสอบทั้งหมดได้ดังตาราง 3.

ตาราง 3. แสดงค่าเฉลี่ยจากการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP

SOAP	ขนาดเนื้อหา		เฉลี่ยจากการทดสอบทั้ง 3 รอบ	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	953	515	24.43	40.95
RETRIEVE	536	1460	23.86	41.92
UPDATE	951	515	23.49	42.58
DELETE	532	515	22.42	44.61

4.2 ผลการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

จากผลการทดสอบของเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นแสดงให้เห็นได้ชัดเจนว่าขนาดของเนื้อหา (Content-Length) ที่ส่งการร้องขอและที่แม่ข่ายเว็บเซอร์วิสตอบกลับมาจะมีขนาดที่เล็กกว่าของ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP อย่างเห็นได้ชัดโดยเฉพาะในกรณีการร้องขอบริการ RETRIEVE และบริการ DELETE นั้นไม่ต้องส่งเนื้อหาใดๆ ไปด้วย เนื่องจากทั้งสองบริการนี้ใช้เพียงการระบุค่าของคีย์หลักซึ่งในที่นี้คือหมายเลขโทรศัพท์ไว้ใน URL เท่านั้น ส่วนขนาดเนื้อหาที่ตอบกลับจากแม่ข่ายเว็บเซอร์วิสในกรณีการใช้บริการ DELETE จะเป็นรหัสแสดงผลลัพธ์ของการลบขนาดประมาณ 67 ไบต์ เนื้อหาของการตอบรับจากบริการ RETRIEVE นั้นจะมีขนาดใหญ่ที่สุดเมื่อเทียบจากทั้ง 4

บริการ เนื่องจากเป็นการตอบกลับข้อมูลสมาชิกด้วยข้อความขนาดประมาณ 316 ไบต์ และกรณีบริการ CREATE และบริการ UPDATE จะมีการส่งการร้องขอที่มีเนื้อหาที่มีข้อมูลสมาชิกขนาดประมาณ 319 ไบต์ และแม่ข่ายเว็บเซอร์วิสจะตอบกลับด้วยผลลัพธ์ของการสร้างและแก้ไขกลับมาด้วยเนื้อหาขนาดประมาณ 67 ไบต์

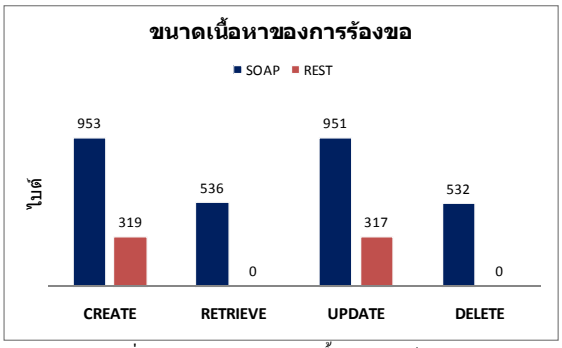
เวลาที่ใช้ในการทำรายการ (Latency) แต่ละรายการเฉลี่ยโดยรวมจากการทดสอบทั้งสามรอบนั้นพบว่าเวลาในการตอบสนองบริการแต่ละครั้งนั้นใช้เวลาอยู่ในช่วง 15-17 มิลลิวินาที และค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาที (Transactions per second) นั้นอยู่ที่ประมาณ 56-62 ทรานแซคชันต่อวินาที โดยบริการ RETRIEVE จะมีค่าปริมาณงานที่ทำในช่วงเวลาหนึ่งวินาทีน้อยที่สุด ดังจะเห็นได้จากตาราง 4.

ตาราง 4. แสดงค่าเฉลี่ยจากการทดสอบเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST

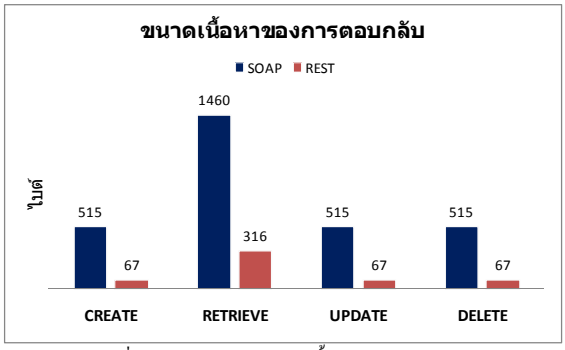
REST	ขนาดเนื้อหา		เฉลี่ยจากการทดสอบทั้ง 3 รอบ	
	การร้องขอ (ไบต์)	การตอบกลับ (ไบต์)	เวลาเฉลี่ย (มิลลิวินาที)	ปริมาณงานที่ทำได้ (ทรานแซคชันต่อวินาที)
CREATE	319	67	16.22	61.68
RETRIEVE	0	316	17.35	57.64
UPDATE	317	67	16.32	62.29
DELETE	0	67	15.91	62.83

จากผลการทดสอบสามารถทำการเปรียบเทียบขนาดของเนื้อหาในการร้องขอบริการจากเว็บเซอร์วิสและขนาดของเนื้อหาในการตอบกลับจากแม่ข่ายเว็บเซอร์วิสตามรูปที่ 7. และ 8. ซึ่งเห็นได้ว่าในการร้องขอบริการ CREATE และบริการ UPDATE จากแม่ข่ายเว็บเซอร์วิสนั้นสามารถลดขนาดของข้อมูลที่ต้องส่งไปได้ประมาณ 66% และการร้องขอบริการ RETRIEVE และบริการ DELETE นั้นสามารถลดขนาดของข้อมูลที่ต้องส่งไปได้ทั้งหมด 100% เพราะด้วยรูปแบบของสถาปัตยกรรมเว็บเซอร์วิสแบบ REST นั้นมีการอ้างอิงคีย์หลักอยู่ใน URL แล้ว จึงทำให้รูปแบบของบริการดังกล่าวไม่ต้องส่งเนื้อหาข้อมูลไปด้วย

และเนื้อหาที่แม่ข่ายเว็บเซอร์วิสตอบกลับนั้นในกรณีที่เป็นการร้องขอข้อมูลด้วยบริการ RETRIEVE นั้นสามารถลดขนาดของเนื้อหาได้ประมาณ 78% และกรณีที่เป็นการร้องขออื่นๆ ที่มีการตอบกลับด้วยรหัสแสดงผลลัพธ์ของการทำรายการต่างๆอย่าง CREATE, UPDATE หรือ DELETE นั้น ถ้าใช้เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST จะสามารถลดขนาดของเนื้อหาตอบกลับได้ประมาณ 87%

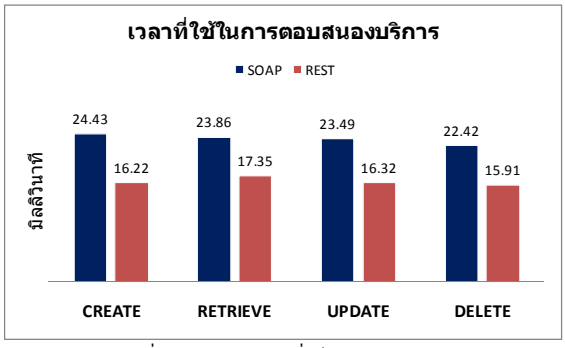


รูปที่ 7. เปรียบเทียบขนาดของเนื้อหาในการร้องขอ



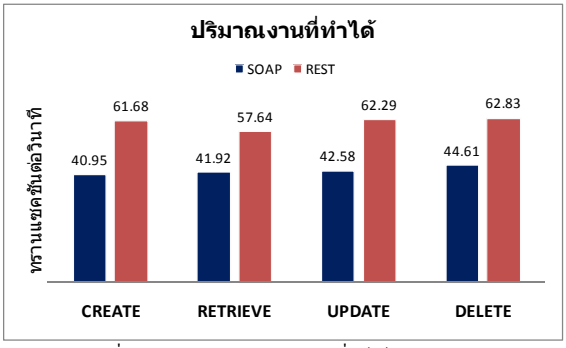
รูปที่ 8. เปรียบเทียบขนาดของเนื้อหาในการตอบกลับ

ในแง่ของเวลาตอบสนองในการใช้บริการ ซึ่งเป็นเวลารวมทั้งหมดที่ใช้ในการส่งการร้องขอบริการไปยังแม่ข่ายเว็บเซอร์วิส เวลาในการประมวลผลของแม่ข่ายเว็บเซอร์วิส และเวลาในการส่งข้อมูลตอบกลับมายังเครื่องลูกข่าย จากข้อมูลเปรียบเทียบเวลาตอบสนองในการใช้บริการในรูปที่ 9. จะเห็นว่าระยะเวลาที่ใช้ในการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST จะน้อยกว่าเวลาตอบสนองของการร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP ประมาณ 30% โดยเฉลี่ยต่อหนึ่งทรานแซคชัน



รูปที่ 9. เปรียบเทียบเวลาที่ใช้ในการทำรายการ

เมื่อเวลาตอบสนองในการร้องขอใช้บริการต่อหนึ่งทรานแซคชันลดลงแล้วนั้นย่อมส่งผลถึงปริมาณงานที่สามารถทำได้ในช่วงเวลาหนึ่งๆ ย่อมเพิ่มขึ้น ดังแสดงให้เห็นการเปรียบเทียบได้จากรูปที่ 10. ซึ่งจะเห็นว่าเมื่อร้องขอใช้บริการเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นมีค่าปริมาณงานที่ทำได้ในช่วงเวลาหนึ่งวินาทีเพิ่มขึ้นประมาณ 30%



รูปที่ 10. เปรียบเทียบค่าปริมาณงานที่ทำได้ในช่วงวินาที

5. บทสรุปงานวิจัย

สำหรับงานวิจัยฉบับนี้ ต้องการชี้ให้เห็นถึงความสำคัญในการเลือกใช้เทคโนโลยีเว็บเซอร์วิสให้เหมาะสมกับความต้องการและลักษณะการใช้งานเพื่อให้เกิดประโยชน์สูงสุดบนทรัพยากรที่มีอยู่จำกัด ในส่วนแรกของงานวิจัยได้กล่าวถึงความรู้เบื้องเกี่ยวกับเว็บเซอร์วิส, SOAP และแนวความคิดเกี่ยวกับการนำทฤษฎีสถาปัตยกรรมแบบ REST มาพัฒนาเป็นเว็บเซอร์วิส ในส่วนที่สองนั้นได้กล่าวถึงวิธีการเรียกใช้บริการ และความแตกต่างของการเรียกใช้บริการ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP กับ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST และในส่วนสุดท้ายจะเป็นการทดสอบเปรียบเทียบให้เห็นถึงความแตกต่างของการเรียกใช้บริการ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ SOAP กับ เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST ในการให้บริการแบบเดียวกันในเชิงของขนาดเนื้อหาที่ใช้ในการร้องขอบริการและการตอบรับ รวมทั้งเวลาในการตอบสนองทั้งหมดของการร้องขอบริการนั้นๆเพื่อเป็นแนวทางในการตัดสินใจเลือกใช้รูปแบบเทคโนโลยีเว็บเซอร์วิสได้อย่างเหมาะสม

จากข้อสรุปผลการทดลองจะเห็นได้ว่าการเรียกใช้งาน เว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นสามารถลดปริมาณข้อมูลการร้องขอบริการและการตอบกลับระหว่างเครื่องลูกข่ายและเครื่องแม่ข่ายได้ 80% โดยประมาณ ในกระบวนการสร้างหรือแก้ไขและลดปริมาณการรับส่งข้อมูลได้มากถึง 90% โดยประมาณ ในกรณีที่เป็นการเรียกดูข้อมูลหรือลบข้อมูลในตำแหน่งทรัพยากรที่ต้องการ ทั้งยังช่วยทำให้เครื่องแม่ข่ายสามารถรองรับการให้บริการเพิ่มมากขึ้นได้โดยที่การร้องขอบริการจากเว็บเซอร์วิสที่ใช้สถาปัตยกรรมแบบ REST นั้นจะช่วยลดระยะเวลาในการตอบสนองการขอบริการ และเพิ่มปริมาณงานที่ทำได้ของแม่ข่ายเว็บเซอร์วิสได้ถึงประมาณ 30% และหากผู้ให้บริการเว็บเซอร์วิสใช้การพัฒนาเว็บเซอร์วิสภายใต้สถาปัตยกรรมแบบ REST ก็จะทำให้ผู้ขอใช้บริการนั้นได้รับการตอบสนองที่เร็วขึ้น และยังเป็นการช่วยทำให้มีการใช้งานแบนด์วิธอย่างคุ้มค่าอีกด้วย

ทั้งนี้การพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตกรรม REST อาจจะมีความยุ่งยากมากกว่าการพัฒนาเว็บเซอร์วิสที่ใช้สถาปัตยกรรม SOAP จึงทำให้เว็บเซอร์วิสที่ใช้สถาปัตกรรม REST ไม่เป็นที่นิยมมากนัก แต่ด้วยข้อดีของการให้บริการด้วยเว็บเซอร์วิสที่ใช้สถาปัตยกรรม REST ดังที่แสดงในผลการทดสอบนั้น ชี้ให้เห็นว่าเว็บเซอร์วิสที่ใช้สถาปัตกรรม REST นั้นเหมาะที่จะให้บริการผู้ใช้บริการที่เชื่อมต่ออินเทอร์เน็ตเครือข่ายที่มีแบนด์วิธไม่สูงมากนัก อย่างเช่นการใช้งาน GPRS/EDGE [9] บนโทรศัพท์มือถือ ซึ่งจะทำให้ผู้ใช้บริการนั้นสามารถใช้บริการได้เร็วขึ้นและยังเป็นการใช้แบนด์วิธอย่างคุ้มค่าอีกด้วย

สำหรับการศึกษารูปแบบการนำทฤษฎีสถาปัตยกรรมแบบ REST มาใช้ให้เต็มประสิทธิภาพและเป็นแนวทางที่ชัดเจนขึ้น ทั้งเรื่องของการกำหนดโครงสร้างต่างๆหรือแม้กระทั่งการประยุกต์ใช้ทฤษฎีสถาปัตยกรรมแบบ REST ไปช่วยในการปรับปรุงระบบที่ใช้งานปัจจุบันอยู่นั้นกำลังอยู่ในช่วงการวิจัยศึกษาค้นคว้า และจัดเตรียมเพื่อจะเผยแพร่ต่อไป

เอกสารอ้างอิง

[1] Roy Thomas Fielding, “Architectural Styles and the Design of Network-based Software Architectures”, Chapter 5, PhD thesis, University of California, Irvine, 2000

[2] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L.Masinter, P.Leach and T.Berners-Lee, “Hypertext transfer protocol 1.1”, Internet, June 1999

[3] Nilo Mitra, Yves Lafon, “SOAP Version 1.2 Part 0: Primer (Second Edition)”, Published on the Internet, W3C Recommendation, April 2007

[4] Alex Rodriguez, “RESTful Web Services: The basics”, IBM, November 2008

[5] T. Berners-Lee, L.Masinter and M. McCahill, “RFC 1738 : Uniform Resource Locators (URL)”, Published: www.rfc-editor.org, December 1994

[6] Martin Gudgin, Marc Hadley and Tony Rogers, “Web Services Addressing 1.0 – Core”, Published on the Internet, W3C Recommendation, May 2006

[7] Roberto Chinnici, Jean-Jacques Moreau, Arthur Ryman, Sanjiva Weerawarana, “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”, Published on the Internet, W3C Candidate Recommendation, March 2006

[8] Naga, Sravanthi (119500) Performance Testing CoE, “Open Source Performance Testing Using Apache JMeter”, Cognizant Technology Solutions, 2008

[9] J. Schiller, “Mobile Communications”, Addison-Wesley, 2000