ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)

ปริญญา

วิทยาการคอมพิวเตอร์ | วิทยาการคอมพิวเตอร์

สาขา | **ภาควิชา**

เรื่อง แนวทางการออกแบบระบบการทำงานสำรองสำหรับเซิร์ฟเวอร์กลุ่มเมฆ โดยใช้แนวคิดแบบไฮบริดจ์คลาวด์

Hybrid Cloud Methodology for Designing Failover System of Cloud Computing

นามผู้วิจัย นายภาคภูมิ มุกดาสนิท

ได้พิจารณาเห็นชอบ

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(อาจารย์เสฏฐวิทย์ เกิดผล, Ph.D.)

หัวหน้าภาควิชา

(ผู้ช่วยศาสตราจารย์สิริกร จันทร์นวล, M.S.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

(รองศาสตราจารย์กัญจนา ธีระกุล, D.Agr.)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

แนวทางการออกแบบระบบการทำงานสำรองสำหรับเซิร์ฟเวอร์กลุ่มเมฆ โดยใช้แนวคิดแบบไฮบริดจ์คลาวด์

Hybrid Cloud Methodology for Designing Failover System of Cloud Computing

โดย

นายภาคภูมิ มุกดาสนิท

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์)
พ.ศ. 2556

ภาคภูมิ มุกดาสนิท 2556: แนวทางการออกแบบระบบการทำงานสำรองสำหรับเซิร์ฟเวอร์กลุ่มเมฆ โดยใช้แนวคิดแบบไฮบริดจ์คลาวด์ ปริญญาวิทยาศาสตรมหาบัณฑิต (วิทยาการคอมพิวเตอร์) สาขาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: อาจารย์เสฏฐวิทย์ เกิดผล, Ph.D. 80 หน้า

เราสามารถนำกลุ่มเมฆปิดส่วนตนมาใช้เป็นระบบการทำงานสำรองแทนในกรณีที่ไม่สามารถใช้บริการจากกลุ่มเมฆสาธารณะ งานวิจัยนี้เราได้ออกแบบกลุ่มเมฆปิดส่วนตนให้เป็นระบบการทำงานสำรองสำหรับกลุ่มเมฆสาธารณะ เนื่องจากกลุ่มเมฆปิดส่วนตนนั้น มีปริมาณทรัพยากรที่น้อยกว่ากลุ่มเมฆสาธารณะ ดังนั้นกลุ่มเมฆปิดส่วนตนจึงไม่สามารถรองรับปริมาณภาระงานทั้งหมดที่เคยทำงานอยู่บนกลุ่มเมฆสาธารณะได้ เราจึงแก้ปัญหาดังกล่าว โดยการปรับจากฟังก์ชั่นการทำงานเต็มที่ใช้งานบนกลุ่มเมฆสาธารณะ ให้เป็นฟังก์ชั่นการทำงานแบบย่อสำหรับกลุ่มเมฆปิดส่วนตน เมื่อเปรียบเทียบกันระหว่างฟังก์ชั่นการทำงานเต็มกับฟังก์ชั่นการทำงานแบบย่อแล้ว ฟังก์ชั่นการทำงานแบบย่อนี้จะใช้ปริมาณทรัพยากรที่น้อยกว่า แต่ก็จะมีคุณภาพในการให้บริการที่ต่ำกว่าเช่นเดียวกัน นอกจากนี้เราได้นิยามรูปแบบการเขียนข้อมูลออกเป็น 3 รูปแบบ คือ ข้อมูลที่ถูกเขียนหลายครั้ง, ข้อมูลที่ถูกเขียนครั้งเดียว และข้อมูลที่ถูกเขียนอย่างเดียวโดยจำแนกตามลักษณะการเปลี่ยนแปลงของข้อมูลนั้น กับลักษณะการนำข้อมูลนั้นไปใช้งานต่อ เราปรับจากฟังก์ชั่นการทำงานเต็มให้เป็นฟังก์ชั่นการทำงานหลักโดยการลดจำนวนรูปแบบการเขียนข้อมูลหรือแปลงจากรูปแบบหนึ่งไปเป็นอีกรูปแบบหนึ่ง รูปแบบการเขียนข้อมูลทั้ง 3 ประเภทจะส่งผลต่อปริมาณภาระงานของระบบที่แตกต่างกัน ผลการทดลองแสดงให้เห็นว่า (1) ระบบที่ใช้ฟังก์ชันการทำงานเต็ม ประกอบด้วย ข้อมูลที่เขียนครั้งเดียว 3 ตัว, ข้อมูลที่ถูกเขียนหลายครั้ง 1 ตัว, และข้อมูลที่ถูกเขียนอย่างเดียว 1 ตัว ระบบจะสามารถรองรับผู้ใช้งานได้เฉลี่ย 469 คน, (2) เมื่อเราปรับจากฟังก์ชันการทำงานเต็มให้เป็นฟังก์ชันการทำงานแบบย่อแบบที่ 1 โดยการแปลงจากข้อมูลที่เขียนครั้งเดียว 3 ตัว ให้กลายเป็นข้อมูลที่ถูกเขียนอย่างเดียวทั้งหมด ระบบจะสามารถรองรับผู้ใช้งานได้เฉลี่ย 572 คน ซึ่งสามารถรองรับผู้ใช้ได้เพิ่มขึ้น 21.96% และ (3) เมื่อเราปรับจากฟังก์ชันการทำงานแบบย่อในข้อ (2) โดยการลบข้อมูลที่ถูกเขียนหลายครั้งออกไป ระบบจะสามารถรองรับผู้ใช้ได้ 1,755 คน ซึ่งสามารถรองรับผู้ใช้ได้เพิ่มขึ้น 206.81%

/ /

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Pakpoom Mookdarsanit 2013: Hybrid Cloud Methodology for Designing Failover System of Cloud Computing. Master of Science (Computer Science),
Major Field: Computer Science, Department of Computer Science.
Thesis Advisor: Mr. Sethavidh Gertphol, Ph.D. 80 pages.

Availability is one of three biggest issues which slow down the adoption of the Public Cloud. A Private Cloud can be designed to use as a replica when Public Cloud goes offline. In this research, we introduce the method for designing Private Cloud as a failover system for Public Cloud. Since the Private Cloud has less resource; it cannot handle all workloads previously on the Public Cloud. We overcome this restriction by adapting the full operation performed in Public Cloud into a light-weight operation optimized for a Private Cloud. A light-weight operation consumes less resource but also has removed quality of service (QoS) compared to the full operation. In other words, we remove the quality of service of the system in order to serve more requests. We also define a write type of data according to change and future use of that data in 3 types: Multiple-write data, Write-once data and Write-only data. We adapt the full operation into a light-weight one by removing the number of write types or changing one type to another. Because these three write-types have different number of workloads. Experimental result shows that, (1) a system with full operation consists of three Write-once data, one Write-only data and one Multiple-write data can sustain an average of 469 maximum numbers of requests. (2) When we adapt full operation into light-weight version by converting all Write-once data into Write-only data, a system can sustain 572 requests, an improvement of 21.96%. (3) When we adapt light-weight operation in (2) by removing a Multiple-write data, a system can sustain 1755 requests, an improvement of 206.81%.

_____ / ____ / ____

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณ อาจารย์เสฏฐวิทย์ เกิดผล ประธานกรรมการที่ปรึกษา ที่ได้ให้คำปรึกษาในด้านการเรียน การค้นคว้าวิจัย ชี้แนะแนวทางแก้ไขปัญหาในงานวิจัย ด้วยความเอาใจใส่ทุกขั้นตอน จนกระทั่งงานวิจัยฉบับนี้เสร็จสมบูรณ์

ขอกราบขอบพระคุณ อาจารย์อุษา สัมมาพันธ์ ที่ได้ให้คำปรึกษา และตำราต่างๆ รวมไปถึงการจัดอบรมการทดสอบประสิทธิภาพของโปรแกรมโดยใช้โปรแกรม JMeter

ขอกราบขอบพระคุณ คณาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ทุกท่าน ที่ได้อบรมสั่งสอนและให้ความรู้อันเป็นประโยชน์เสมอมา

ขอขอบคุณ คุณนพธน ผาสุก นิสิตปริญญาโท สาขาวิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ ที่ได้พัฒนาเครื่องมือย่อยของซอฟต์แวร์ JMeter สำหรับมาใช้ในการทดสอบ

ด้วยความดีหรือประโยชน์อันใดเนื่องจากวิทยานิพนธ์เล่มนี้ ขอมอบแด่บิดา มารดา ครูอาจารย์ และผู้มีพระคุณทุกท่านที่ได้อบรมและให้กำลังใจผู้วิจัยมาตลอดในทุกเรื่อง

ภาคภูมิ มุกดาสนิท

มกราคม 2556

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

แนวทางการออกแบบระบบการทำงานสำรองสำหรับเซิร์ฟเวอร์กลุ่มเมฆ โดยใช้แนวคิดแบบไฮบริดจ์คลาวด์

Hybrid Cloud Methodology

for Designing Failover System of Cloud Computing

คำนำ

การประมวลผลแบบกลุ่มเมฆ เป็นรูปแบบหนึ่งในการเข้าใช้ทรัพยากรคอมพิวเตอร์ผ่านทางระบบเครือข่ายโดยผู้ใช้ไม่จำเป็นต้องทราบรายละเอียดทางด้านฮาร์ดแวร์ เพราะทรัพยากรฮาร์ดแวร์ทั้งหมดจะถูกใช้งานร่วมกันระหว่างผู้ใช้ โดยทางฝั่งผู้ให้บริการ จะคอยจัดสรรทรัพยากรแบบเสมือน และแจกจ่ายทรัพยากรตามปริมาณการใช้งานของผู้ใช้แต่ละคน ซึ่งผู้ใช้แต่ละคนสามารถควบคุมโครงสร้างทรัพยากรเสมือนของตนได้โดยไม่ต้องผ่านบุคลากรทางฝั่งผู้ให้บริการ การเปลี่ยนแปลงลักษณะโครงสร้างระบบแบบเดิมมาเป็นโครงสร้างระบบแบบเสมือนนี้จะช่วยลดต้นทุนสำหรับการจัดการทางด้านฮาร์ดแวร์และซอฟต์แวร์ เพราะเซิร์ฟเวอร์กลุ่มเมฆจะจัดสรรทรัพยากรฮาร์ดแวร์และ/หรือซอฟต์แวร์ในรูปแบบบริการ ซึ่งสามารถปรับขนาดได้ตามความเหมาะสมตามปริมาณงาน และคิดค่าบริการตามปริมาณทรัพยากรที่มีผู้ใช้ได้ใช้งานจริง

ความแตกต่างระหว่างกลุ่มเมฆสาธารณะ และกลุ่มเมฆปิดส่วนตน คือ กลุ่มเมฆสาธารณะจะใช้ทรัพยากรเสมือนจากผู้ให้บริการภายนอก เมื่อองค์กรใช้บริการจากกลุ่มเมฆสาธารณะ ทรัพยากรต่างๆจะถูกใช้งานร่วมกันกับผู้ใช้คนอื่นๆที่อยู่ภายนอกองค์กร ส่วนกลุ่มเมฆปิดส่วนตนเป็นโครงสร้างเสมือนที่ถูกสร้างขึ้นไว้ใช้สำหรับผู้ที่เกี่ยวข้องกับองค์กรเท่านั้น ที่จะมีสิทธิ์ใช้งานทรัพยากรดังกล่าวได้ ถ้าระบบขององค์กรใดใช้บริการทั้งกลุ่มเมฆปิดส่วนตนและกลุ่มเมฆปิดสาธารณะจะเรียกว่ากลุ่มเมฆแบบผสมผสาน

สำหรับการใช้บริการจากกลุ่มเมฆสาธารณะนั้น ปัญหาทางด้านความพร้อมในการให้บริการ เป็นหนึ่งในสามปัญหาที่นักวิจัยให้ความสำคัญมากที่สุด องค์กรที่ใช้บริการกลุ่มเมฆสาธารณะ จะต้องสามารถสื่อสารกับผู้ให้บริการ เพื่อแลกเปลี่ยนข้อมูลกันผ่านทางสัญญาณเครือข่าย ดังนั้น ทางฝั่งผู้ใช้งานและทางฝั่งผู้ให้บริการจะต้องสามารถเชื่อมต่อกันได้อยู่ตลอดเวลา ระบบจึงจะสามารถทำงานได้ อาจมีบางเหตุการณ์ที่ทำให้ไม่สามารถทำงานบนกลุ่มเมฆสาธารณะได้

ตามปกติ ยกตัวอย่างเช่น เมื่อเดือนมิถุนายน พ.ศ. 2555 ที่ผ่านมา เกิดพายุซัดทำให้กลุ่มเมฆสาธารณะของ Amazon ไม่สามารถให้บริการได้ ในช่วงเวลาหนึ่ง ซึ่งส่งผลให้องค์กรต่างๆ ที่ใช้บริการกลุ่มเมฆสาธารณะนี้ เช่น Netflix, Pinterrest และ Instagram หยุดให้บริการไปโดยอัตโนมัติ จึงมีความจำเป็นจะต้องกำหนดแผนรับมือกับปัญหาต่างๆที่อาจจะเกิดขึ้นเมื่อไม่สามารถเข้าถึงกลุ่มเมฆสาธารณะได้ตามปกติ เพราะถึงแม้ว่ากลุ่มเมฆสาธารณะอยู่ในสภาวะไม่พร้อมในการให้บริการแล้ว แต่ระบบขององค์กรจะต้องยังคงดำเนินการทางธุรกิจต่อไปได้ ท่ามกลางสภาวะที่ล้มเหลวของกลุ่มเมฆสาธารณะ

การนำเซิร์ฟเวอร์กลุ่มเมฆปิดส่วนตนมาใช้เป็นระบบการทำงานสำรองสำหรับกลุ่มเมฆสาธารณะนั้น เป็นเทคนิคหนึ่งที่สามารถใช้แก้ปัญหาดังกล่าวเพื่อให้ระบบยังคงสามารถดำเนินต่อไปได้ท่ามกลางสภาวะล้มเหลวของกลุ่มเมฆสาธารณะ กล่าวคือ เมื่อกลุ่มเมฆสาธารณะไม่สามารถใช้งานได้ตามปกติ ก็จะสลับการทำงานทั้งหมดมาที่กลุ่มเมฆปิดส่วนตน และการออกแบบระบบการทำงานสำรองในกลุ่มเมฆปิดส่วนตนก็เป็นปัญหาอีกประเด็นหนึ่ง เพราะกลุ่มเมฆปิดส่วนตนนั้นมีปริมาณทรัพยากรที่จำกัด ทำให้กลุ่มเมฆปิดส่วนตนจึงไม่สามารถรองรับปริมาณภาระงานทั้งหมดบนกลุ่มเมฆสาธารณะได้ เราแก้ปัญหาดังกล่าวโดยการปรับจากฟังก์ชันการทำงานเต็มบนกลุ่มเมฆสาธารณะให้เป็นฟังก์ชันการทำงานแบบย่อสำหรับกลุ่มเมฆปิดส่วนตน เราได้นิยามฟังก์ชั่นการทำงานเต็ม คือ ความสามารถในการทำงานของระบบตามความต้องการของผู้ใช้ ส่วนฟังก์ชั่นการทำงานแบบย่อ คือ ลดความสามารถในการทำงานของระบบตามความต้องการของผู้ใช้บางส่วนออกไป ซึ่งทำให้ฟังก์ชันการทำงานแบบย่อนั้นมีคุณภาพในการให้บริการต่ำกว่าฟังก์ชันการทำงานเต็ม เพราะมีบริการที่น้อยกว่า ซึ่งจะทำให้มีปริมาณภาระงานที่น้อยกว่าและรองรับจำนวนผู้ใช้ได้มากขึ้น นอกจากนี้เรายังนิยามรูปแบบการเขียนข้อมูลของฟังก์ชันการทำงานเต็มและฟังก์ชันการแบบย่อออกเป็น 3 ประเภทตามรูปแบบการเปลี่ยนข้อมูล และการนำข้อมูลไปใช้งานต่อ คือ (1) ข้อมูลที่ถูกเขียนหลายครั้ง, (2) ข้อมูลที่ถูกเขียนครั้งเดียว และ (3) ข้อมูลที่ถูกเขียนอย่างเดียว ข้อมูลประเภทที่ (1) และ (2) นั้น จำเป็นจะต้องมีการอ่านข้อมูลใดๆ เพื่อคำนวณก่อนการเขียนข้อมูล และภายหลังจากข้อมูลนั้นมีการเปลี่ยนแปลง จะมีการนำไปใช้งานต่อในอนาคต ข้อมูลประเภทที่ (1) จะถูกนำไปใช้งานต่อในอนาคตสำหรับการอ่าน/เขียน ส่วนข้อมูลประเภทที่ (2) จะถูกนำไปใช้งานต่อในอนาคตสำหรับการอ่านอย่างเดียว แต่ข้อมูลประเภทที่ (3) นั้นอาจจะไม่จำเป็นต้องมีการอ่านข้อมูลใดๆก่อนการเขียนข้อมูล และภายหลังจากข้อมูลนั้นมีการเปลี่ยนแปลง จะไม่มีการนำไปใช้งานต่อในการอ่านหรือเขียนอนาคต เราใช้วิธีการปรับจากฟังก์ชันการทำงานเต็มให้กลายเป็นฟังก์ชันการทำงานแบบย่อสำหรับระบบการทำงานสำรองบนกลุ่มเมฆปิดส่วนตน ทำได้ 3 วิธี คือ (1) แปลงจากข้อมูลที่เขียนครั้งเดียว ให้กลายเป็นข้อมูลที่ถูกเขียนอย่างเดียว, (2) ลบข้อมูล

ประเภทอื่นทิ้ง ให้เหลือไว้แต่รูปแบบการเขียนข้อมูลประเภท ข้อมูลที่ถูกเขียนอย่างเดียว และ (3) ลบข้อมูลที่ถูกเขียนอย่างเดียว ซึ่งเกินความจำเป็นออกไป

ระบบการทำงานสำรองจะมี 3 สภาวะ คือ สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ, สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน, และสภาวะคืนข้อมูลจากกลุ่มเมฆปิดส่วนตนกลับสู่กลุ่มเมฆสาธารณะ

สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ ถือว่าเป็นสภาวะที่ระบบจะทำงานตามปกติ เมื่อมีการเปลี่ยนแปลงของข้อมูลบางประเภทเกิดขึ้นบนกลุ่มเมฆสาธารณะ อาจจะต้องมีการสำเนาข้อมูลไปยังกลุ่มเมฆปิดส่วนตน เพื่อใช้งานในสภาวะที่กลุ่มเมฆสาธารณะไม่สามารถให้บริการได้

เมื่อระบบไม่สามารถใช้งานบนกลุ่มเมฆสาธารณะได้ ก็จะสลับมาทำงานที่ระบบการทำงานสำรองบนกลุ่มเมฆปิดส่วนตน เรียกว่า สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน และเนื่องจากปริมาณทรัพยากรบนกลุ่มเมฆส่วนตนน้อยกว่าบนกลุ่มเมฆสาธารณะ ดังนั้นจึงจำเป็นจะต้องปรับฟังก์ชันการทำงานเต็มบนกลุ่มเมฆสาธารณะให้เป็นฟังก์ชันการทำงานแบบย่อที่เหมาะสมกับกลุ่มเมฆปิดส่วนตน ซึ่งจะต้องปรับโดยคำนึงถึงคุณภาพให้อยู่ในเกณฑ์ที่พอรับได้ ในส่วนของรูปแบบการเขียนข้อมูล ทั้ง 3 รูปแบบ ก็มีลักษณะที่แตกต่างกัน ซึ่งแต่ละรูปแบบจะใช้ปริมาณงานและเวลาที่ไม่เท่ากันเช่นเดียวกัน

เมื่อกลุ่มเมฆสาธารณะกลับมาใช้งานได้ตามปกติ ก็จะมีการส่งค่าข้อมูลที่มีการเปลี่ยนแปลงภายในกลุ่มเมฆปิดส่วนตนทั้งหมดกลับไปยังกลุ่มเมฆสาธารณะ เรียกว่า สภาวะคืนข้อมูลกลับสู่กลุ่มเมฆสาธารณะ ในส่วนของรูปแบบการเขียนข้อมูลแต่ละรูปแบบ ก็จะมีลักษณะการคืนข้อมูลที่แตกต่างกัน ซึ่งแต่ละรูปแบบจะใช้ปริมาณงานและเวลาที่ไม่เท่ากันเช่นเดียวกัน

งานวิจัยของเราจะทดสอบเฉพาะสภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตนเท่านั้น และผลการทดลองแสดงให้เห็นว่า (1) ระบบที่ใช้ฟังก์ชันการทำงานเต็ม ประกอบด้วย ข้อมูลที่เขียนครั้งเดียว 3 ตัว, ข้อมูลที่ถูกเขียนหลายครั้ง 1 ตัว, และข้อมูลที่ถูกเขียนอย่างเดียว 1 ตัว ระบบจะสามารถรองรับผู้ใช้งานได้เฉลี่ย 469 คน, (2) เมื่อเราปรับจากฟังก์ชันการทำงานเต็มให้เป็นฟังก์ชันการทำงาน แบบย่อ โดยการแปลงจากข้อมูลที่เขียนครั้งเดียว 3 ตัว ให้กลายเป็นข้อมูลที่ถูกเขียนอย่างเดียวทั้งหมด ระบบจะสามารถรองรับผู้ใช้งานได้เฉลี่ย 572 คน ซึ่งสามารถรองรับผู้ใช้ได้เพิ่มขึ้น 21.96% และ (3) เมื่อเราปรับจากฟังก์ชันการทำงานแบบย่อในข้อ (2) โดยการลบข้อมูลที่ถูกเขียน

หลายครั้งออกไป ระบบจะสามารถรองรับผู้ใช้ได้ 1,755 คน ซึ่งสามารถรองรับผู้ใช้ได้เพิ่มขึ้น 206.81% ดังนั้น เมื่อเราปรับฟังก์ชันการทำงานเต็มของระบบให้เป็นฟังก์ชันการทำงานแบบย่อ จะทำให้ระบบสามารถรองรับจำนวนผู้ใช้ได้เพิ่มขึ้น หรืออีกนัยหนึ่งคือ เราลดคุณภาพในการให้บริการ เพื่อเพิ่มประสิทธิภาพของระบบในการรองรับจำนวนผู้ใช้ได้มากยิ่งขึ้น

วัตถุประสงค์

1. เพื่อปรับจากฟังก์ชันการทำงานเต็ม เป็นฟังก์ชันการทำงานแบบย่อ โดยใช้วิธีการแปลง และ/หรือ ลดรูปแบบการเขียนข้อมูลของฟังก์ชันการทำงาน

2. เพื่อลดปริมาณงานของฟังก์ชันการทำงาน ซึ่งส่งผลต่อการรองรับจำนวนผู้ใช้ของระบบได้มากยิ่งขึ้น

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ปรับจากฟังก์ชันการทำงานเต็มเป็นฟังก์ชันการทำงานแบบย่อสำหรับระบบการจัดการข้อมูลสารสนเทศแบบอื่นๆ โดยใช้วิธีการแปลง และ/หรือ ลดรูปแบบการเขียนข้อมูลของฟังก์ชันการทำงาน

2. ได้ลดปริมาณงานสำหรับฟังก์ชันการทำงาน และระบบสามารถรองรับจำนวนผู้ใช้ได้มากยิ่งขึ้น

ขอบเขตและข้อจำกัด

1. นิยามฟังก์ชันการทำงานและรูปแบบการเขียนข้อมูลของฟังก์ชันการทำงาน 3 รูปแบบ

2. ระบบที่ใช้เป็นกรณีศึกษา คือระบบการจองตั๋วภาพยนตร์ 3 รูปแบบเป็นระบบสำรองบนกลุ่มเมฆปิดส่วนตน โดยแต่ละรูปแบบมีฟังก์ชันการทำงานแตกต่างกัน

3. ทำการทดสอบระบบทั้ง 3 รูปแบบ โดยการทำ Load Testing เพื่อวัดประสิทธิภาพระบบการจองตั๋วทั้ง 3 รูปแบบของฟังก์ชันการทำงานแบบย่อ

4. ทำการทดสอบระบบทั้งหมดเฉพาะระบบการทำงานสำรอง บนสภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตนเท่านั้น

การตรวจเอกสาร

ทฤษฎีที่เกี่ยวข้องกับงานวิจัย

การประมวลผลแบบกลุ่มเมฆ

เทคโนโลยีการประมวลผลแบบกลุ่มเมฆได้เปลี่ยนรูปแบบโครงสร้างทรัพยากรรูปแบบเดิม คือ ทรัพยากรต่างๆจะถูกรวมไว้ที่ศูนย์กลาง และผู้ใช้จะแบ่งปันกันใช้ทรัพยากร โดยผู้ใช้ไม่จำเป็นต้องทราบที่ตั้งหรือรายละเอียดต่างๆของทรัพยากร แต่ผู้ใช้สามารถร้องขอไปยังเครือข่ายศูนย์กลางทรัพยากร ซึ่งรับคำสั่งจากผู้ใช้และส่งผลลัพธ์กลับมาแสดงผลผ่านทางสัญญาณเครือข่าย

1. ลักษณะของเซิร์ฟเวอร์กลุ่มเมฆ

เป็นรูปแบบหนึ่งในการเข้าใช้ทรัพยากรคอมพิวเตอร์ผ่านทางระบบเครือข่าย ซึ่งผู้ใช้สามารถระบุจำนวนทรัพยากร โดยไม่ต้องคำนึงถึงปัญหาต่างๆในระดับฮาร์ดแวร์ เช่น ปริมาณความจุ, การใช้พลังงาน เป็นต้น รูปแบบการให้บริการกลุ่มเมฆที่คนส่วนใหญ่คุ้นเคย คือ Software as a service (SaaS) ยกตัวอย่างเช่น Facebook, Google Drive, iCloud เป็นต้น ทรัพยากรต่างๆของเซิร์ฟเวอร์กลุ่มเมฆจะถูกรวมเข้าไว้ด้วยกันที่แหล่งศูนย์กลางและผู้ใช้จะแบ่งปันกันใช้ทรัพยากรต่างๆเหล่านี้ เช่น เครื่องแม่ข่าย, หน่วยเก็บข้อมูล, แอพลิเคชั่น, หรือบริการอื่นๆ เป็นต้น เนื่องจากการใช้บริการจากเซิร์ฟเวอร์กลุ่มเมฆนี้ ผู้ใช้สามารถปรับขนาดทรัพยากรได้ และระบบจะจัดสรรทรัพยากรให้โดยอัตโนมัติ ซึ่งสามารถใช้งานได้ทันที โดยไม่ต้องพึ่งพาบุคลากรของผู้ให้บริการ ทำให้การจัดสรรทรัพยากรเป็นไปได้อย่างรวดเร็ว คุณลักษณะของตัวแบบกลุ่มเมฆนี้ประกอบไปด้วย 5 ประการ ที่เป็นประโยชน์ต่อผู้ใช้ดังนี้

1.1 การจัดสรรทรัพยากรให้ผู้ใช้โดยอัตโนมัติ คือ เมื่อทางฝั่งผู้ใช้ร้องขอหน่วยทรัพยากรจากเซิร์ฟเวอร์กลุ่มเมฆ เช่น หน่วยความเก็บข้อมูลของเครือข่าย, เวลาของเครื่องแม่ข่าย เป็นต้น ระบบจะจัดสรรทรัพยากรให้ผู้ใช้โดยอัตโนมัติ โดยไม่ต้องผ่านบุคลากรของผู้ให้บริการ

1.2 ความสามารถในการใช้บริการจากอุปกรณ์อื่นๆ คือ ผู้ให้บริการกลุ่มเมฆยังคงให้บริการทรัพยากรกับผู้ใช้ผ่านทางระบบเครือข่ายได้ ไม่ว่าผู้ใช้จะเข้ามาใช้บริการผ่านทางอุปกรณ์ใดๆ เช่น โทรศัพท์มือถือ, คอมพิวเตอร์ส่วนตัว เป็นต้น

1.3 แหล่งทรัพยากรที่ใช้งานร่วมกัน คือ ผู้ให้บริการกลุ่มเมฆจัดสรรแหล่งทรัพยากรเพื่อแจกจ่ายไปยังผู้ใช้ตามปริมาณการใช้งาน ผู้ใช้จะใช้ทรัพยากรร่วมกัน แต่จะไม่สามารถทราบรายละเอียดที่ตั้งของแหล่งทรัพยากร และไม่สามารถควบคุมการจัดสรรทรัพยากรของระบบได้

1.4 ความยืดหยุ่นในการให้บริการ คือ ความยืดหยุ่นในการให้บริการนับว่าเป็นคุณสมบัติหนึ่ง คือ จำนวนทรัพยากรฮาร์ดแวร์สามารถปรับได้อย่างไม่จำกัดตามปริมาณที่ต้องการใช้งาน เช่น ถ้ามหาวิทยาลัยหนึ่งไปใช้บริการเซิร์ฟเวอร์กลุ่มเมฆจากภายนอก โดยมีข้อกำหนดในการใช้บริการ เช่น มีการระบุไว้อย่างชัดเจนว่าใช้ทรัพยากรจำนวนหนึ่ง แต่เมื่อถึงวันลงทะเบียนเรียน อาจมีนิสิตเข้ามาใช้บริการเป็นจำนวนมาก ในวันนั้นระบบอาจจำเป็นจะต้องปรับจำนวนทรัพยากรเพื่อให้สามารถรองรับจำนวนผู้ใช้เพิ่มขึ้น เป็นต้น

1.5 มาตรวัดจำนวนการใช้บริการ คือ ผู้ให้บริการกลุ่มเมฆจะต้องสามารถปรับขนาดทรัพยากรให้เหมาะสมกับการใช้งานของผู้ใช้ได้เองโดยอัตโนมัติ จะมีการวัดจำนวนการใช้งานทรัพยากร เช่น หน่วยประมวลผล, หน่วยความจำหลัก เป็นต้น และระบบจะต้องสามารถตรวจดูควบคุมปริมาณทรัพยากรที่เหมาะสมกับการใช้งานของผู้ใช้ และออกเป็นรายงานข้อมูลการใช้งานทั้งหมดได้

2. รูปแบบการนำไปใช้งานจริง

2.1 กลุ่มเมฆสาธารณะ (Public Cloud) เป็นโครงสร้างกลุ่มเมฆที่เปิดเพื่อให้ผู้ใช้ทั่วไปเข้ามาใช้บริการได้ เช่น องค์กรธุรกิจ สถานศึกษา องค์กรภาครัฐ เป็นต้น ที่ตั้งของโครงสร้างทรัพยากรอยู่ที่ฝั่งผู้ให้บริการ

2.2 กลุ่มเมฆปิดส่วนตน (Private Cloud) เป็นโครงสร้างกลุ่มเมฆถูกกำหนดขึ้นไว้ใช้งานเฉพาะทางสำหรับองค์กรหนึ่ง ผู้ใช้จะต้องเป็นเกี่ยวข้องกับองค์กรนั้นถึงจะมีสิทธิ์ใช้งานได้ ที่ตั้งของโครงสร้างทรัพยากรอยู่ที่ฝั่งองค์กรเอง หรือตั้งไว้ภายนอกองค์กรก็ได้

2.3 กลุ่มเมฆแบบผสมผสาน (Hybrid Cloud) เป็นโครงสร้างของกลุ่มเมฆตั้งแต่ 2 กลุ่มใดๆ ที่ใช้มาตรฐานเดียวกันในการทำงานร่วมกัน หรือแลกเปลี่ยนข้อมูลร่วมกัน

3. รูปแบบการให้บริการของเซิร์ฟเวอร์กลุ่มเมฆ

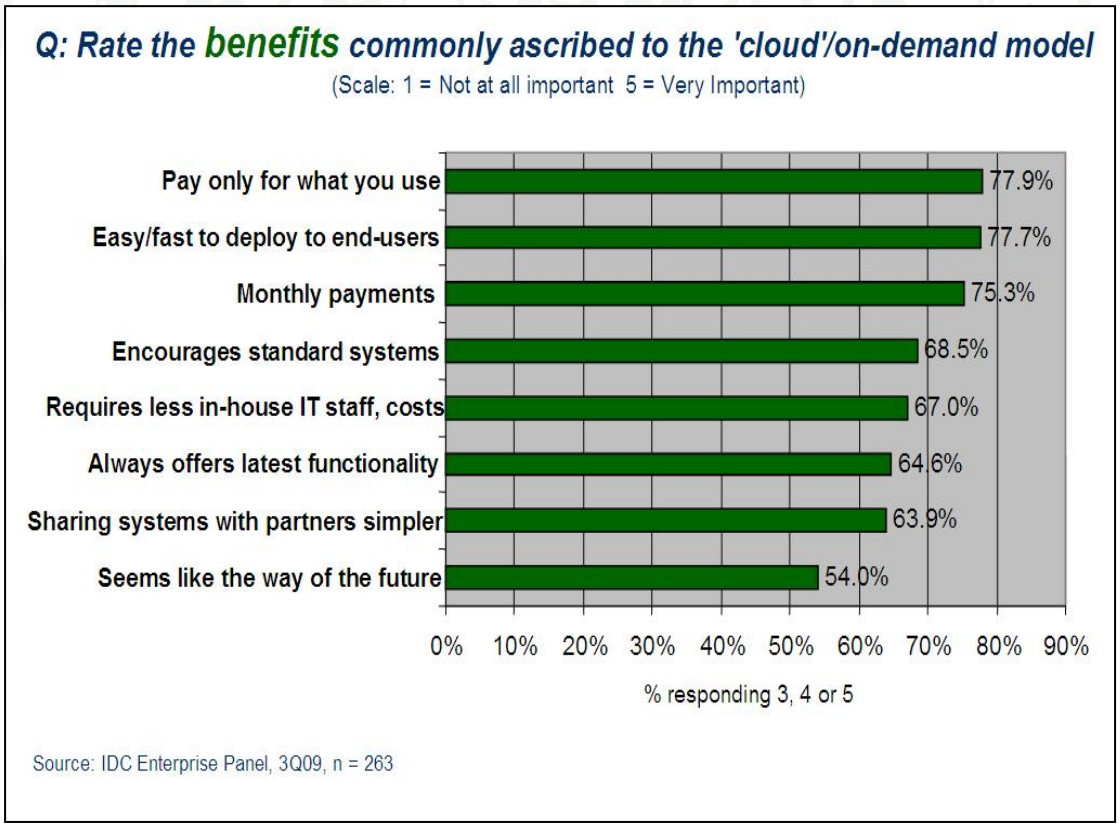
3.1 Infrastructure as a Service (IaaS) เป็นรูปแบบบริการที่ผู้ให้บริการได้จัดสรรโครงสร้างทรัพยากรเสมือน เช่น หน่วยประมวลผล, หน่วยความจำหลัก, หน่วยเก็บข้อมูล และฮาร์ดแวร์อื่นๆให้กับผู้ใช้ โดยผู้ใช้สามารถนำซอฟต์แวร์ใดๆไม่ว่าจะเป็นระบบปฏิบัติการหรือโปรแกรมมาทำงานบนโครงสร้างทรัพยากรเหล่านี้ แต่ผู้ใช้สามารถเข้าถึงโครงสร้างทรัพยากรได้เพียงบางส่วนเท่านั้น เช่น Instagram ไปใช้บริการเช่าเครือข่ายทางด้านฮาร์ดแวร์เสมือนจาก Amazon เพื่อรองรับจำนวนผู้ใช้จำนวนมากๆได้ แต่ระบบที่ใช้ในการจัดการไฟล์รูปภาพต่างๆได้อย่างรวดเร็วนั้น เป็นระบบที่ Instagram พัฒนาขึ้นเอง

3.2 Platform as a Service (PaaS) เป็นรูปแบบบริการที่ผู้ให้บริการอนุญาตให้ผู้ใช้นำระบบใดๆมาใช้งานบนโครงสร้างกลุ่มเมฆได้ ผู้ให้บริการจัดสรรไลบรารี่, ภาษาที่ใช้ในการเขียนโปรแกรม หรือบริการอื่นๆ ให้ผู้ใช้สามารถเขียนโปรแกรมเพื่อใช้งานบนกลุ่มเมฆได้ แต่ผู้ใช้ไม่สามารถเข้าถึงโครงสร้างของกลุ่มเมฆได้ เช่น ทรัพยากรระบบเครือข่าย, หน่วยเก็บข้อมูล, ระบบปฏิบัติการ เป็นต้น ผู้ใช้จะมีสิทธิ์แค่นำบริการต่างๆที่ผู้ให้บริการได้จัดสรรไว้ ไปใช้สร้างระบบเพื่อใช้งานบนโครงสร้างกลุ่มเมฆได้ เช่น สถาบันการศึกษาแห่งหนึ่งสามารถนำระบบลงทะเบียนเรียนของนิสิตไปทำงานบน OpenStack ได้

3.3 Software as a Service (SaaS) เป็นรูปแบบบริการที่ผู้ให้บริการจัดสรรทรัพยากรซอฟต์แวร์ที่ทำงานอยู่บนแหล่งทรัพยากรฮาร์ดแวร์บนกลุ่มเมฆ ผู้ใช้สามารถใช้งานซอฟต์แวร์ผ่านทางอุปกรณ์ต่างๆหรือเว็บบราวเซอร์ได้ แต่ผู้ใช้ไม่สามารถเข้าถึงโครงสร้างของกลุ่มเมฆได้ เช่น ทรัพยากรระบบเครือข่าย, หน่วยเก็บข้อมูล, ระบบปฏิบัติการ เป็นต้น ผู้ใช้จะมีสิทธิ์ใช้งานซอฟต์แวร์ได้อย่างเดียว ยกเว้นผู้ใช้ที่ได้รับสิทธิ์เท่านั้น ยกตัวอย่างเช่น Google เปิดให้ผู้ใช้สามารถเข้าไป Google Doc เพื่อจัดการเกี่ยวกับเอกสาร

4. ประโยชน์ที่ผู้ใช้จะได้รับเมื่อเลือกใช้บริการกลุ่มเมฆ

จากภาพที่ 1 แสดงถึงประโยชน์ที่ผู้ใช้จะได้รับเมื่อใช้บริการจากกลุ่มเมฆ ซึ่งรวบรวมโดย IDC ปี 2009 จากผู้ประเมินทั้งหมด 263 คน โดยทาง IDC ได้กำหนดประโยชน์ที่ผู้ใช้จะได้รับมา 8 ข้อ และผู้ประเมินให้คะแนนกับประโยชน์ในแต่ละข้อ ซึ่งมีเกณฑ์ตั้งแต่1-5 คะแนน (1 คือสำคัญน้อย ถึง 5 คือสำคัญมาก) แล้วจำนวนผู้ที่ให้คะแนน 3-5 ในแต่ละข้อ มาคิดเป็นเปอร์เซ็นต์จากจำนวนผู้ประเมินทั้งหมด และเปรียบเทียบเป็นกราฟดังภาพที่ 1 จะเห็นได้ว่าประโยชน์ส่วนใหญ่ที่ผู้ใช้มองเห็นประโยชน์ทางด้านค่าใช้จ่าย คิดเป็น 77.9% เพราะบริการของกลุ่มเมฆจะช่วยลดค่าใช้จ่ายสำหรับผู้ใช้ กล่าวคือ ผู้ใช้เสียค่าใช้จ่ายเฉพาะปริมาณทรัพยากรที่มีการใช้งานทางฝั่งกลุ่มเมฆเท่านั้น ซึ่งประหยัดต้นทุนทางด้านการวางโครงสร้างของระบบและการดูแลรักษา ประโยชน์รองลงมา คือ สามารถติดตั้งและนำไปใช้งานได้ง่าย

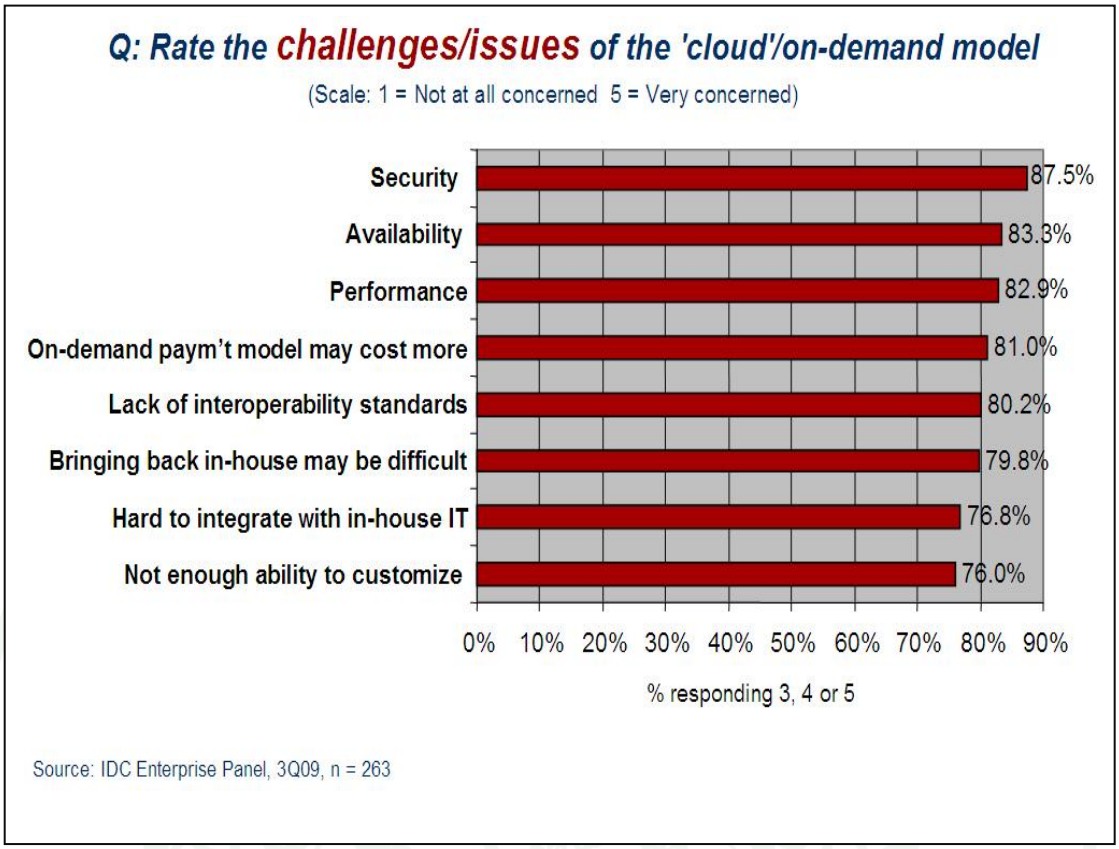


ภาพที่ 1 เหตุผลที่ผู้ใช้เลือกใช้บริการกลุ่มเมฆสาธารณะ

ที่มา: IDC Cloud (2009)

5. ปัญหาในประเด็นต่างๆ สำหรับการใช้บริการเซิร์ฟเวอร์กลุ่มเมฆ

จากภาพที่ 2 แสดงถึงประเด็นปัญหาที่เกิดจากการนำกลุ่มเมฆมาใช้งาน ซึ่งรวบรวมโดย IDC ปี 2009 จากผู้ประเมินทั้งหมด 263 คน โดยทาง IDC ได้กำหนดประเด็นปัญหาที่เกิดขึ้นจากการใช้งานกลุ่มเมฆมา 8 ข้อ และผู้ประเมินให้คะแนนกับปัญหาในแต่ละข้อ ซึ่งมีเกณฑ์ตั้งแต่1-5 คะแนน (1 คือ เป็นผลกระทบน้อย ถึง 5 คือ เป็นผลกระทบอย่างมาก) แล้วนำจำนวนผู้ที่ให้คะแนน 3-5 ในแต่ละข้อมาคิดเป็นเปอร์เซ็นต์จากจำนวนผู้ประเมินทั้งหมด และเปรียบเทียบเป็นกราฟดังภาพที่ 2 จะเห็นได้ว่า 3 อันดับแรก ที่ผู้ใช้มองว่าเป็นประเด็นปัญหาอย่างมากสำหรับการใช้งานกลุ่มเมฆ คือ Security, Availability และ Performance สำหรับประเด็นทางด้าน Security คือ เนื่องจากข้อมูลทุกอย่างถูกเก็บอยู่ที่ฝั่งผู้ให้บริการกลุ่มเมฆ ซึ่งเป็นการแบ่งปันหน่วยเก็บข้อมูลในการเก็บข้อมูลร่วมกันระหว่างผู้ใช้คนอื่น ผู้ใช้ไม่มีทางทราบได้เลยว่าการจัดการของระบบในการเก็บข้อมูลลงหน่วยเก็บข้อมูลมีความปลอดภัยมากน้อยเพียงใด, ประเด็นทางด้าน Availability คือ เนื่องจากการทำงานทุกอย่างจะใช้ทรัพยากรทางฝั่งผู้ให้บริการ แต่เมื่อทางฝั่งผู้บริการไม่สามารถให้บริการได้ตามปกตินั้น จะทำให้ระบบงานของผู้ใช้ไม่สามารถใช้งานได้เช่นกัน, ประเด็นทางด้าน Performance คือ เนื่องจากการประมวลผลทุกอย่างอยู่บนฝั่งผู้ให้บริการและผลลัพธ์จะถูกส่งไปยังเครื่องของผู้ใช้ ดังนั้นจะต้องใช้เวลาในการส่งข้อมูลระหว่างผู้ใช้และผู้ให้บริการทุกครั้ง



ภาพที่ 2 ประเด็นปัญหาสำหรับกลุ่มเมฆ

ที่มา: IDC Cloud (2009)

ความคงทนของระบบ

คือ คุณสมบัติของระบบ ที่ทำให้ระบบสามารถทำงานต่อไปได้ แม้ว่ามีองค์ประกอบบางอย่างผิดปกติ หรืออีกความหมายหนึ่งเป็นชื่อเรียกของวิธีการจัดการกับความผิดพลาดที่เกิดขึ้นเพื่อให้ระบบสามารถทำงานต่อไปได้ รวมไปถึงการป้องกันความล้มเหลวของการทำงานทั้งหมดของระบบเท่าที่จะสามารถทำได้ โดยทั่วไปแล้วการเกิดข้อผิดพลาดเป็นสิ่งที่หลีกเลี่ยงไม่ได้สำหรับโปรแกรมขนาดใหญ่ จะเน้นการลดการสูญเสียประสิทธิภาพของความล้มเหลวของระบบให้มากที่สุด เราสามารถแบ่งข้อผิดพลาด ได้เป็น 2 ประเภท คือ เกิดความล้มเหลวที่เซิร์ฟเวอร์ศูนย์กลาง หรือ เกิดความล้มเหลวที่โหนดใดโหนดหนึ่ง

1. การทำสำเนาข้อมูล

เป็นเทคนิคการการจัดการข้อผิดพลาดแบบหนึ่ง ในระบบที่มีข้อมูลที่มีการเปลี่ยนแปลงอยู่ตลอดเวลาและกระจายตัว จะมีการโยนงานจากเซิร์ฟเวอร์ตัวที่ล้มเหลวไปยังเซิร์ฟเวอร์ที่สำรองข้อมูลไว้ เพื่อป้องกันการสูญหายของข้อมูล จะกล่าวถึงเทคนิค 2 ประเภท

1.1 การทำสำเนาข้อมูลเชิงรับ เป็นวิธีการจัดการกับความผิดพลาดของระบบ จะอาศัยข้อมูลสำรองในการแก้ไขข้อผิดพลาดข้อมูล และมีองค์ประกอบสำรองขึ้นมาทำงานแทน

1.2 การทำสำเนาข้อมูลเชิงรุก ประกอบด้วย 4 ส่วน คือ

1.2.1 การตรวจสอบความผิดพลาด เป็นกระบวนการที่ทำการตรวจสอบว่ามีความเสียหายเกิดขึ้นหรือไม่ มักเป็นกระบวนการที่เกิดขึ้นกระบวนการแรก ก่อนเข้าสู่กระบวนการอื่น

1.2.2 การระบุตำแหน่งความผิดพลาด เป็นกระบวนการสำหรับหาตำแหน่งของความเสียหายของระบบ เพื่อที่จะสามารถดำเนินการกู้คืนได้

1.2.3 การจำกัดขอบเขตความผิดพลาด เป็นกระบวนการเพื่อแยกความเสียหายออกจากการทำงานของระบบ เพื่อไม่ให้ความเสียหายดังกล่าวส่งผลกระทบต่อส่วนอื่นๆ

1.2.4 การกู้คืนจากความเสียหาย เป็นกระบวนการที่ทำให้ระบบสามารถทำงานได้ต่อเนื่อง หรือกลับมาทำงานได้ ระหว่างการจัดรูปแบบการทำงาน เมื่อเกิดความเสียหายขึ้น

2. ความแตกต่างระหว่างสภาพพร้อมใช้งานของระบบกับความเชื่อมั่นของระบบ

2.1 สภาพร้อมใช้งานของระบบ เป็นคุณสมบัติของระบบที่สามารถทำงานให้บริการต่อผู้ใช้ต่อไปได้ สามารถเพิ่มคุณสมบัตินี้โดย Live Migration

2.2 ความเชื่อมั่นของระบบ เป็นคุณสมบัติของระบบในการเก็บรักษาข้อมูลให้คงอยู่และความถูกต้องของข้อมูล สามารถเพิ่มคุณสมบัตินี้โดย Consolidated Backup

การทดสอบซอฟต์แวร์

เมื่อเราสร้างซอฟต์แวร์เสร็จแล้ว จะต้องมีการทดสอบเพื่อพิสูจน์ว่าซอฟต์แวร์ที่เราสร้างขึ้นมานั้นสามารถทำงานได้ดีมากน้อยเพียงใดก่อนนำไปใช้งานจริง สามารถแบ่งเป็น 2 ประเภท โดยแบ่งตามคุณสมบัติภายในและคุณสมบัติภายนอกของซอฟต์แวร์

1. Functional Testing

เป็นการทดสอบคุณสมบัติภายในของซอฟต์แวร์ ว่าซอฟต์แวร์สามารถทำงานได้ครบทุกฟังก์ชันตามข้อกำหนดหรือไม่ และ/หรือทำงานอย่างถูกต้องหรือไม่

2. Non-functional Testing

เป็นการทดสอบการทดสอบคุณสมบัติภายนอกของซอฟต์แวร์อื่นๆ ในด้านคุณภาพ, ประสิทธิภาพ, และปริมาณ ซึ่งอกเหนือจากคุณสมบัติของซอฟต์แวร์ที่เราออกแบบ เช่น เวลาในการทำงาน, ปริมาณฮาร์ดแวร์ที่ใช้ในการทำงาน, ความคงทนต่อความผิดพลาด เป็นต้น ยกตัวอย่างเช่น

2.1 Stress Testing เป็นการทดสอบโดยหาวิธีการทำให้ซอฟต์แวร์เกิดข้อผิดพลาด ยกตัวอย่างเช่น ใส่ค่าข้อมูลแปลกๆ เพื่อทดสอบว่าความคงทนของซอฟต์แวร์ว่ายังคงสามารถทำงานต่อไปได้หรือไม่

2.2 Performance Testing เป็นการทดสอบว่าซอฟต์แวร์นั้น มีประสิทธิภาพในการทำงานมากน้อยขนาดไหน เช่น ใช้ซีพียูเท่าใด, รองรับจำนวนผู้ใช้ได้ขนาดไหน เป็นต้น

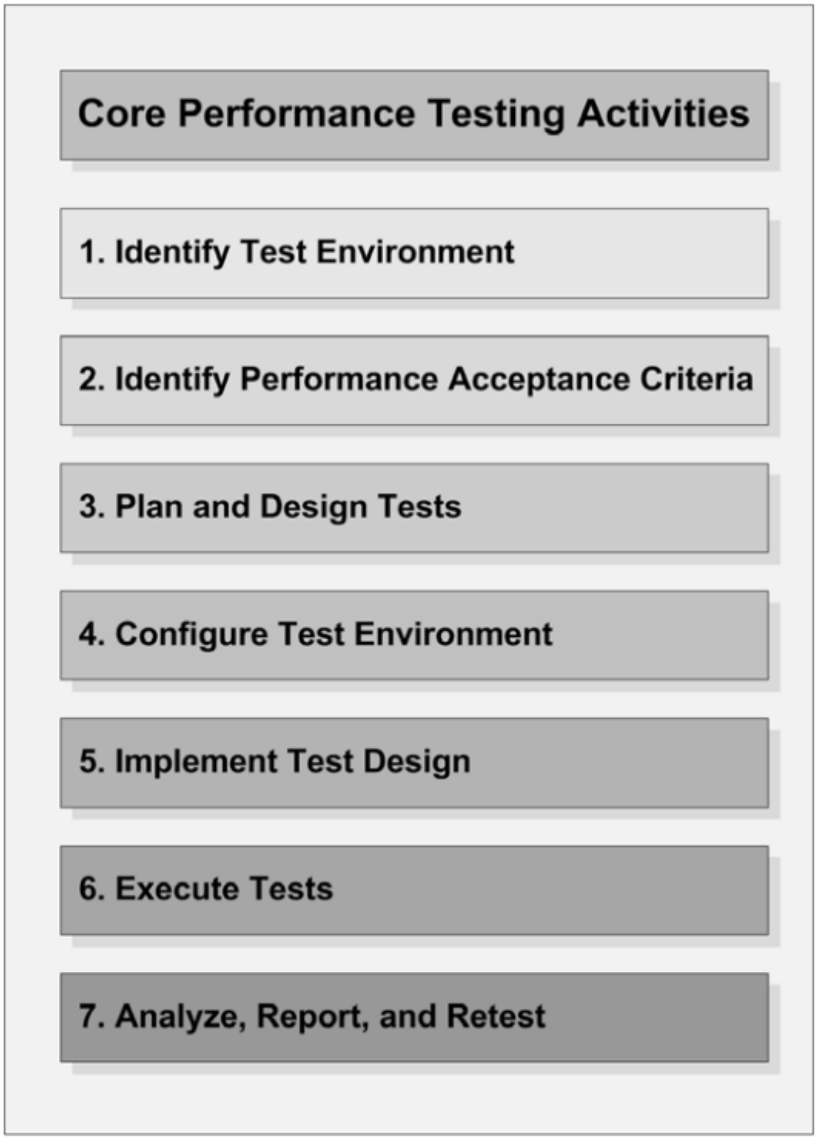
2.3 Endurance Testing เป็นการทดสอบซอฟต์แวร์ภายใต้สภาวะปกติ เป็นเวลานานๆ เพื่อทดสอบดูว่าส่งผลกระทบต่ออะไรต่อสภาวะแวดล้อมบ้าง เช่น ซอฟต์แวร์นี้สามารถทำงานได้อย่างถูกต้องและครบถ้วน แต่เมื่อใช้ไปนานๆแล้ว เกิด Leak Memory (อาจจะเป็นเพราะโปรแกรมเมอร์ไม่มีการคืนหน่วยความจำให้กับระบบปฏิบัติการเป็นต้น)

2.4 Load Testing เป็นการทดสอบว่าระบบสามารถรองรับจำนวนผู้ใช้ได้เท่าใด

การทดสอบประสิทธิภาพของซอฟต์แวร์

เนื่องจากคำว่า “ประสิทธิภาพ” นั้นเป็นคำที่มีความหมายกว้างเกินไป ก่อนที่เราจะวัดประสิทธิภาพของซอฟต์แวร์เพื่อตัดสินว่าซอฟต์แวร์ตัวใดมีประสิทธิภาพมากกว่ากันนั้น เราจำเป็นจะต้องระบุให้ชัดเจนว่าประสิทธิภาพทางด้านใด เช่น ใช้หน่วยความจำในการทำงานมากขนาดไหน, ใช้เวลาในการตอบสนองนานเท่าใด, รองรับผู้ใช้ได้มากที่สุดเท่าใดจึงไม่เกิดข้อผิดพลาด เป็นต้น ซึ่งไม่ว่าจะเป็น Endurance Testing, Isolation Testing, Stress Testing และ Load Testing ตามที่กล่าวมาข้างต้นก็จัดว่าเป็นการทดสอบประสิทธิภาพ (Performance Testing) รูปแบบหนึ่งด้วยเช่นกัน

1. ขั้นตอนการทดสอบประสิทธิภาพ 7 ขั้นตอน



ภาพที่ 3 กระบวนการทดสอบประสิทธิภาพ 7 ขั้นตอน

ที่มา: Microsoft Press (2007)

1.1 กำหนดสภาวะแวดล้อมในการทดสอบ เป็นการกำหนดสภาวะแวดล้อมทั้งหมดที่เกี่ยวข้อง และส่งผลต่อผลการทดลอง ซึ่งได้รวมถึงรายละเอียดโครงสร้างทางด้านเครือข่าย, ฮาร์ดแวร์, และซอฟต์แวร์ที่ใช้ในการทดสอบ และใช้งานจริง ว่ามีกระบวนการทำงานอย่างไร ในการกำหนดนั้นจะต้องมองเห็นองค์รวมทุกอย่างของสภาวะแวดล้อมในการทำงาน ถึงจะวางแผนการทดลอง และออกแบบการทดลองได้อย่างมีประสิทธิภาพ

1.2 กำหนดเงื่อนไขที่สามารถยอมรับได้ เป็นการกำหนดเงื่อนไขต่างๆ เช่น เวลาในการตอบสนอง, ปริมาณการใช้ซีพียู, จำนวนผู้ใช้ที่สามารถรองรับได้, มีข้อผิดพลาดได้ไม่เกินร้อยละเท่าใด เป็นต้น โดยเมื่อเรากำหนดค่าเหล่านี้ไว้แล้ว ระบบจะทำงานโดยในสภาวะที่ค่าของตัวแปรต่างๆเหล่านี้จะไม่เกินตามค่าที่ได้กำหนดไว้ ถ้าเกินเงื่อนไขต่างๆที่เรากำหนดไว้ ระบบก็จะหยุดการทำงาน และนำค่าที่เราต้องการทดสอบมาเปรียบเทียบกัน ณ สภาวะที่หยุดการทำงาน

1.3 วางแผนและออกแบบการทดลอง ในการทดสอบประสิทธิภาพ เราจะวัดประสิทธิภาพด้วยวิธีใด ต้องการทดสอบอะไร ทำการทดสอบอย่างไร ใช้ข้อมูลอะไรในการทดสอบ และใช้ตัวแปรใดบ้างในการวัดประสิทธิภาพ และมีวิธีการเปรียบเทียบประสิทธิภาพอย่างไร

1.4 ติดตั้งสภาวะแวดล้อมในการทดสอบ เตรียมติดตั้งระบบปฏิบัติการ, ข้อมูลที่จะนำมาทดสอบ และซอฟต์แวร์ที่ใช้ทดสอบ เตรียมให้พร้อมที่จะทดสอบ สภาวะแวดล้อมที่ติดตั้งสามารถเป็นมาตรวัดได้อย่างชัดเจน และมองเห็นผลลัพธ์ได้ชัดเจน

1.5 กำหนดค่าต่างๆ ของสภาวะแวดล้อมที่ใช้ในการทดลอง ตามการทดลองที่ได้ออกแบบไว้ เตรียมติดตั้งระบบปฏิบัติการ, ข้อมูลที่จะนำมาทดสอบ และซอฟต์แวร์ที่ใช้ทดสอบ เตรียมให้พร้อมที่จะทดสอบ สามารถระบุและสามารถมองเห็นผลลัพธ์ได้ชัดเจน

1.6 ทำการทดลอง ดำเนินการทดลองตามที่ได้ออกแบบการทดลองไว้ โดยที่ผลลัพธ์ที่ได้จากการทดลองจะต้องไม่ขัดกับเงื่อนไขที่สามารถยอมรับได้ที่ได้กำหนดไว้แล้ว

1.7 วิเคราะห์ผลการทดลอง ทำการบันทึกผลการทดลอง พร้อมทั้งเปรียบเทียบประสิทธิภาพและ/หรือคุณภาพของผลลัพธ์ พร้อมทั้งวิเคราะห์ผลลัพธ์ว่ามีที่มาที่ไปอย่างไรโดยหาเหตุผลที่ได้จากการตั้งสมมติฐานมาทำการวิเคราะห์ หรือเพื่อที่จะทำการทดลองในครั้งต่อไป เพื่อที่สามารถระบุสาเหตุที่แท้จริง เพื่อมาพิสูจน์สมมติฐานที่ได้ตั้งไว้ ซึ่งการตั้งสมมติฐานไว้ก่อนการทดลอง อาจไม่ได้ผลลัพธ์ที่สอดคล้องก็เป็นได้

งานวิจัยที่เกี่ยวข้อง

ในส่วนนี้นำเสนองานวิจัยที่เกี่ยวข้องในเรื่องของวิธีการจัดการความผิดพลาดของระบบแบบต่างๆ Peery *et al.* (2006) นำเสนอวิธีการเพิ่ม Availability ของอุปกรณ์สื่อสาร โดยลดปริมาณงานสำหรับเซิร์ฟเวอร์ส่วนกลางของระบบ โดยการแบ่ง Availability เป็น 3 รูปแบบ คือ Ownership Availability คือ ผู้ใช้สามารถใช้บริการจากข้อมูลซึ่งถูกเก็บไว้ในอุปกรณ์ของผู้ใช้เอง, Offline Availability คือ ผู้ใช้สามารถใช้บริการจากข้อมูลซึ่งถูกเก็บไว้ยังโครงสร้างเครือข่ายที่อุปกรณ์ของผู้ใช้ได้ทำการเชื่อมต่ออยู่, และ Online Availability คือ ผู้ใช้สามารถใช้บริการจากข้อมูล ซึ่งเก็บอยู่บนเซิร์ฟเวอร์ส่วนกลางของระบบ หลักแนวคิด คือ ใช้หลักการแบบเดียวกับ Cache คือ เรียกข้อมูลที่ถูกเก็บอยู่ใกล้ผู้ใช้ที่สุด บริการต่อผู้ใช้ เพื่อความรวดเร็วในการบริการ Yang *et al.* (2006) ทำการออกแบบเครื่องมือที่เรียกว่า PRIN (ย่อมาจาก Optimizing Performance of Reliable Internet Storage) ในการทำ Replication เพื่อจะช่วยลด Traffic จำนวนมหาศาลของข้อมูลสำเนาที่ถูกเก็บบนระบบเครือข่าย และยังช่วยลดเวลาในการตอบสนองอีกด้วย วิธีการนี้ใช้แนวคิดคล้ายๆ RAID กล่าวคือ แทนที่จะสำเนาทั้ง Data Block ของข้อมูลขณะเขียนแต่ละครั้ง ก็จะสำเนาเพียงแค่ค่า Parity ของ Data Block และส่งไปตามระบบเครือข่าย และ Data Block จะถูกคำนวณออกมาทีหลังจากที่ได้รับค่า Parity แล้ว ซึ่งเป็นวิธีการแก้ปัญหาในระดับ Bit level Zorzo *et al.* (2005) นำเสนอตัวแบบสำหรับระบบเพื่อเพิ่มความคงทนของระบบ เป็นการวางแผนรับมือเมื่อระบบทำงานผิดพลาด โดยการทดสอบยิงข้อผิดพลาดหลายๆรูปแบบต่างๆที่ถูกจำลองขึ้นไปยังระบบ แล้วทำการบันทึกผลการทดลอง ว่าในการยิงข้อผิดพลาดแต่ละรูปแบบนั้นมี Node ใดที่ได้รับผลกระทบบ้าง เพื่อทำการสำเนาข้อมูลของ Node นั้นไว้กับ Node ที่อยู่ข้างเคียง Bora *et al.* (2005) ออกแบบตัวแบบระบบการจัดการความผิดพลาดสำหรับกลุ่มเซิร์ฟเวอร์แบบกระจายตัว โดยระบบนี้จะมีกลุ่มเครือข่ายหลายๆ กลุ่ม แต่ละกลุ่มเครือข่ายจะมี Node ตัวแทน ทำหน้าคอยจัดสรรทรัพยากรสำหรับสมาชิกภายในกลุ่ม เมื่อ Node ใดทำงานผิดพลาด ก็จะทำการจัดการสำเนาข้อมูลมายัง Node อื่นๆ และทำหน้าติดต่อสื่อสารกับตัวแทนที่อยู่ในกลุ่มเครือข่ายอื่นๆทั้งหมด เพื่อทำการสำเนาข้อมูลร่วมกันอย่างเป็นระบบ

ในส่วนนี้นำเสนองานวิจัยที่เกี่ยวข้องในเรื่องของการลดคุณภาพในการให้บริการของระบบแบบต่างๆ Abdelzaher *et al.* (1999) นำเสนอวิธีการปรับปริมาณงานของเว็บเซิร์ฟเวอร์ โดยการปรับคุณภาพในการให้บริการ เพื่อลดรายละเอียดต่างๆของเว็บที่แสดงผลต่อผู้ใช้ โดยมีวิธีต่างๆ คือ ลดความละเอียดของรูปภาพที่แสดงผลต่อผู้ใช้, ลดจำนวนวัตถุต่างๆที่มาแสดงบนเว็บ เช่น รูปภาพ,

เสียง เป็นต้น และลดจำนวน Link ภายในเว็บ Gopshtein *et al.* (2011) นำแนวคิดของ Abdelzaher and และ Bhatti มาต่อยอดโดยเพิ่มการทำการปรับคุณภาพในการให้บริการระหว่างคุณภาพต่ำและคุณภาพสูงในการแสดงผลต่อผู้ใช้โดยอัตโนมัติ โดยที่ระบบจะใช้คุณภาพตามปกติ ในขณะที่ระบบรองรับจำนวนผู้ใช้ไม่เยอะมาก เมื่อผู้ใช้เข้ามาใช้งานจำนวนมาก ระบบจะลดคุณภาพความคมชัดและความละเอียดของไฟล์วีดีโอนั้นในการให้บริการ เพื่อให้เว็บเซิร์ฟเวอร์สามารถรองรับปริมาณงานได้อย่างเหมาะสม Schwarz *et al.* (2007) ได้สรุปงานวิจัยทั้งหมดที่เกี่ยวข้องกับการปรับคุณภาพความคมชัดและความละเอียดของไฟล์วีดีโอ ใช้วิธีการลดคุณภาพเพื่อให้การขนส่งไฟล์ที่รวดเร็วและสะดวกยิ่งขึ้นเช่น bit rate, format และ power adaptation

อุปกรณ์และวิธีการ

อุปกรณ์

1. ฮาร์ดแวร์ (Hardware):

1.1 คอมพิวเตอร์ Core 2 Duo, RAM 2 GB DDR2

1.2 คอมพิวเตอร์ AMD E-450, RAM 2GB

2. ซอฟต์แวร์ (Software)

2.1 Microsoft Windows Vista Ultimate Service Pack 1 (32-bit)

2.2 Netbeans-jee-galileo-win32

2.3 Java Development Kit (JDK) jdk-6u23-windows-i586

2.4 Apache-jmeter-addins

2.5 Apache-tomcat-6.0 (6.0.35)

วิธีการ

การออกแบบกลุ่มเมฆปิดส่วนตนเพื่อใช้เป็นระบบการทำงานสำรองสำหรับกลุ่มเมฆสาธารณะนั้น เป็นการนำแนวคิดกลุ่มเมฆแบบผสมผสาน มาช่วยในการแก้ปัญหาดังกล่าว ซึ่งจะมีกลุ่มเมฆหลักอยู่ 2 กลุ่มเมฆ คือ กลุ่มเมฆสาธารณะ และกลุ่มเมฆปิดส่วนตน ในสภาวะปกติการดำเนินงานฟังก์ชันการทำงานเต็มจะอยู่บนกลุ่มเมฆสาธารณะ แต่เมื่อไม่สามารถเข้าไปใช้งานกลุ่มเมฆสาธารณะได้นั้น ก็จะเข้ามาทำงานที่กลุ่มเมฆปิดส่วนตน เนื่องจากกลุ่มเมฆปิดส่วนตนมีปริมาณทรัพยากรที่น้อยกว่า จึงไม่สามารถรองรับปริมาณงานทั้งหมดที่ทำอยู่บนกลุ่มเมฆสาธารณะได้ เราจึงแก้ปัญหานี้โดยการปรับฟังก์ชันการทำงานเต็มบนกลุ่มเมฆสาธารณะให้เป็นฟังก์ชันการทำงานแบบย่อสำหรับบนกลุ่มเมฆปิดส่วนตน

ตัวแบบของระบบ

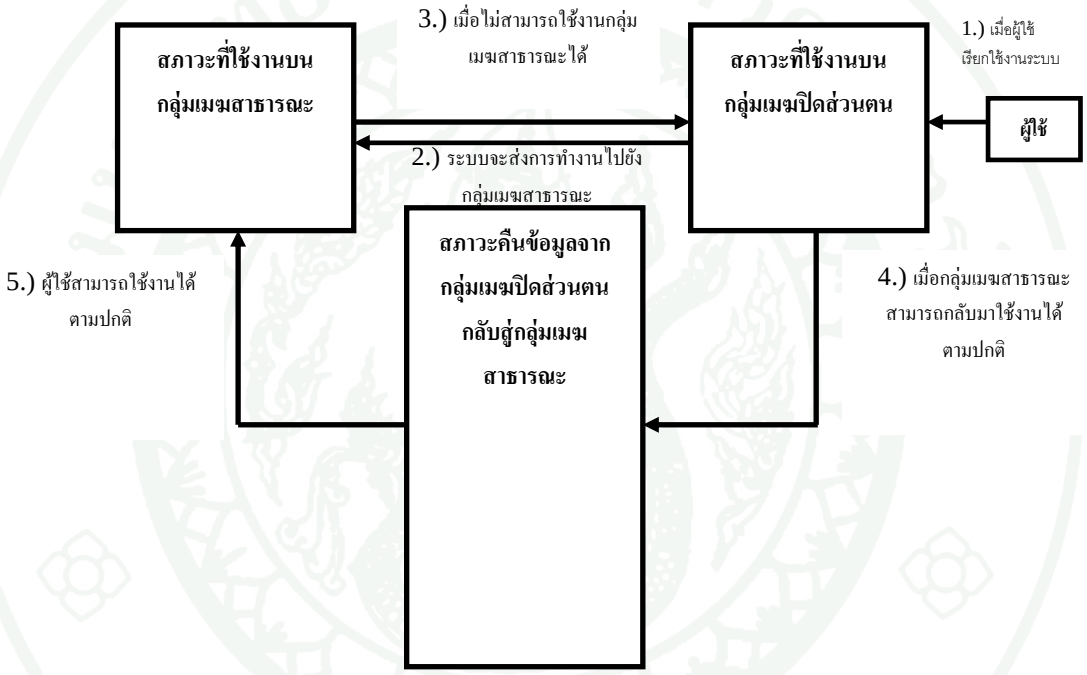
การทำงานของระบบนั้นจะมี 3 สภาวะ คือ สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ, สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน, และสภาวะคืนข้อมูลจากกลุ่มเมฆปิดส่วนตนกลับสู่กลุ่มเมฆสาธารณะ

สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ ถือว่าเป็นสภาวะที่ระบบจะทำงานตามปกติ คือขั้นตอนแรก ผู้ใช้จะเรียกใช้งานที่กลุ่มเมฆปิดส่วนตนก่อน หลังจากนั้นกลุ่มเมฆปิดส่วนตนจะเป็นผู้ส่งการทำงานมายังกลุ่มเมฆสาธารณะ เมื่อมีการเปลี่ยนแปลงของข้อมูลบางประเภทเกิดขึ้นบนกลุ่มเมฆสาธารณะ อาจจะต้องมี: (1) การทำสำเนาข้อมูลเพื่อไปเก็บยังกลุ่มเมฆปิดส่วนตนด้วย และ/หรือ (2) อาจจะมีการตั้งเวลาทุกๆหน่วยเวลา เช่น ทุกๆ 15 นาทีมีการสำเนากลุ่มข้อมูลที่มีการเปลี่ยนแปลงไปยังกลุ่มเมฆปิดส่วนตน และ/หรือ (3) ในกรณีที่ข้อมูลที่มีการเปลี่ยนแปลงแล้วไม่มีการนำไปใช้ต่อ ก็ไม่จำเป็นจะต้องสำเนาข้อมูลที่มีการเปลี่ยนแปลงไปยังกลุ่มเมฆปิดส่วนตน จาก (1) - (3) เพื่อให้ข้อมูลที่มีการเปลี่ยนแปลงล่าสุด สามารถนำมาใช้ต่อบนกลุ่มเมฆปิดส่วนตนได้ เมื่อไม่สามารถเข้าถึงกลุ่มเมฆสาธารณะ

เมื่อระบบไม่สามารถใช้งานบนกลุ่มเมฆสาธารณะได้ ก็จะสลับมาทำงานที่ระบบการทำงานสำรองบนกลุ่มเมฆปิดส่วนตน เรียกว่า สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน และเนื่องจากปริมาณทรัพยากรบนกลุ่มเมฆส่วนตนน้อยกว่าบนกลุ่มเมฆสาธารณะ ดังนั้นจึงจำเป็นจะต้องปรับฟังก์ชันการทำงานเต็มบนกลุ่มเมฆสาธารณะให้เป็นฟังก์ชันการทำงานแบบย่อที่เหมาะสมกับกลุ่ม

เมฆปิดส่วนตน ซึ่งจะต้องปรับโดยคำนึงถึงคุณภาพให้อยู่ในเกณฑ์ที่พอรับได้ และจะมีหน่วยตรวจสอบอยู่ตลอดเวลาว่าสามารถใช้งานกลุ่มเมฆสาธารณะได้หรือยัง

เมื่อกลุ่มเมฆสาธารณะกลับมาใช้งานได้ตามปกติ ก็จะมีการส่งค่าข้อมูลที่มีการเปลี่ยนแปลงที่เกิดขึ้นในขณะที่ใช้งานกลุ่มเมฆปิดส่วนตนทั้งหมด กลับไปยังกลุ่มเมฆสาธารณะ เรียกว่า สภาวะคืนข้อมูลกลับสู่กลุ่มเมฆสาธารณะ ลักษณะการเปลี่ยนแปลงของข้อมูลแต่ละแบบก็จะมีลักษณะการคืนข้อมูลกลับไปยังกลุ่มเมฆสาธารณะที่แตกต่างกัน



ภาพที่ 4 ตัวแบบของระบบการทำงานสำรอง

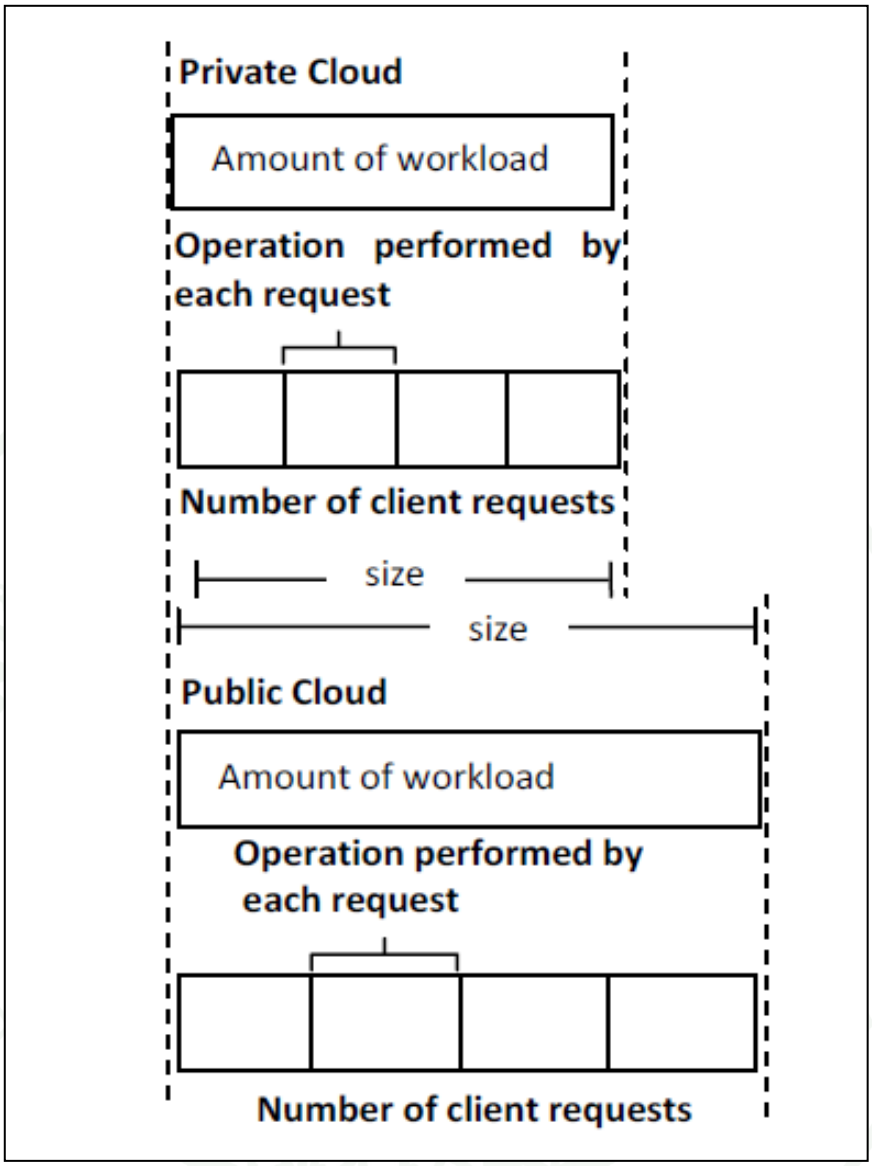
งานวิจัยนี้ศึกษาการทำงานของระบบในสภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตนเท่านั้น

ปริมาณภาระงานของระบบ (Workload)

ปริมาณงานของระบบขึ้นอยู่กับ 2 ปัจจัย คือ จำนวนผู้ใช้ และฟังก์ชันการทำงานที่บริการต่อผู้ใช้ เรานิยามฟังก์ชันการทำงานเต็ม ว่าเป็นการทำงานของโปรแกรมตามความต้องการของระบบ ยกตัวอย่างเช่น ฟังก์ชันการทำงานเต็มของการจองตั๋วภาพยนตร์ ประกอบไปด้วย (1) ตรวจสอบว่ามีจำนวนที่นั่งที่ยังว่างหรือไม่, (2) ตรวจสอบว่าหมายเลขที่นั่งที่เราต้องการนั้นถูกจองแล้วหรือยัง, (3) ทำการจองตั๋วพร้อมกับหมายเลขที่นั่ง, และ (4) ลดจำนวนที่นั่งที่ยังว่างไป 1 ที่นั่ง

เราได้นิยาม ฟังก์ชันการทำงานแบบย่อ คือ การทำงานของโปรแกรมโดยตัดความต้องการของระบบบางอย่างออกไป ยกตัวอย่างเช่น จากตัวอย่างที่แล้ว ฟังก์ชันการทำงานแบบย่อของการจองตั๋วภาพยนตร์ คือ ไม่มีการตรวจสอบว่ามีจำนวนที่นั่งที่ยังว่างหรือไม่ แต่จะเป็นการรับฝากข้อมูลการจองตั๋วไว้กับระบบ และไม่มีการลดจำนวนที่นั่งที่ยังว่างลงไป 1 ที่นั่ง เมื่อเวลาผ่านไประบบจะทำการจองตั๋วให้เองโดยอัตโนมัติตามคิว ระบบจะไม่มีการรับประกันว่าผู้ใช้สามารถจองตั๋วได้ เพราะเสมือนเป็นเพียงการรับฝากเรื่องเท่านั้น ซึ่งฟังก์ชันการทำงานแบบย่อจะมีคุณภาพในการให้บริการต่ำกว่าฟังก์ชันการทำงานเต็ม

รูปภาพที่ 5 แสดงให้เห็นว่า กลุ่มเมฆสาธารณะนั้นมีความจุ และมีปริมาณงานที่มากกว่ากลุ่มเมฆปิดส่วนตน เมื่อเซิร์ฟเวอร์กลุ่มเมฆสาธารณะไม่สามารถให้บริการได้ ปริมาณงานจะถูกลดเพื่อสลับเข้ามาทำงานยังกลุ่มเมฆปิดส่วนตน จากที่ได้กล่าวข้างต้น ปริมาณงานขึ้นกับ 2 ปัจจัย คือ จำนวนผู้ใช้ กับฟังก์ชันการทำงานของระบบ เราสามารถลดปริมาณงานได้โดยการลดจำนวนผู้ใช้ หรือปรับจากฟังก์ชันการทำงานเต็มให้กลายเป็นฟังก์ชันการทำงานแบบย่อ เนื่องจากจำนวนผู้ใช้เป็นปัจจัยภายนอกที่เราไม่สามารถควบคุมได้โดยตรง การเปลี่ยนรูปแบบของฟังก์ชันการทำงานจึงเป็นวิธีที่ง่ายกว่า เมื่อเราปรับจากฟังก์ชันการทำงานเต็ม เป็นฟังก์ชันการทำงานแบบย่อสำหรับกลุ่มเมฆปิดส่วนตน ทำให้ฟังก์ชันการทำงานบนกลุ่มเมฆปิดส่วนตนจะมีบริการที่น้อยกว่าบนกลุ่มเมฆสาธารณะ แต่จะใช้ปริมาณงานในการประมวลผลที่น้อยกว่า ซึ่งทำให้กลุ่มเมฆปิดส่วนตนสามารถรองรับจำนวนผู้ใช้ได้เท่ากับกลุ่มเมฆสาธารณะ หรืออีกนัยหนึ่ง คือ เราลดคุณภาพในการให้บริการของระบบเพื่อที่จะให้ระบบสามารถรองรับจำนวนผู้ใช้ได้มากยิ่งขึ้น



ภาพที่ 5 ลักษณะเปรียบเทียบปริมาณงานระหว่างกลุ่มเมฆสาธารณะและกลุ่มเมฆปิดส่วนตน

ผลกระทบของการลดคุณภาพในการให้บริการ

เราปรับจากฟังก์ชันการทำงานเต็มเป็นฟังก์ชันการทำงานแบบย่อเพื่อลดปริมาณงานของกลุ่มเมฆปิดส่วนตนและเพิ่มจำนวนผู้ใช้ อย่างไรก็ตาม ระบบที่ใช้ฟังก์ชันการทำงานแบบย่อก็จะเป็นการลดคุณภาพในการให้บริการต่อผู้ใช้ด้วยเช่นกัน เพราะบริการต่างๆที่ตอบสนองต่อผู้ใช้ก็จะกลายเป็นบริการที่ไม่สมบูรณ์ จาก 2 ตัวอย่างที่แล้ว ฟังก์ชันการทำงานเต็มของระบบการจองตั๋วภาพยนตร์ คือ สามารถรับประกันกับผู้ใช้ได้ว่า ผู้ใช้ได้จองตั๋วภาพยนตร์อย่างสมบูรณ์แล้ว มีที่นั่งแน่นอน แต่ฟังก์ชันการทำงานแบบย่อของระบบจองตั๋วจะมีการลดคุณภาพในการให้บริการ คือ ระบบได้รับเรื่องการจองตั๋วจากผู้ใช้ และการจองตั๋วจะถูกดำเนินการโดยระบบทีหลังตามลำดับ ซึ่งไม่สามารถรับประกันต่อผู้ใช้ได้ว่ามีที่นั่งแน่นอน เป็นต้น

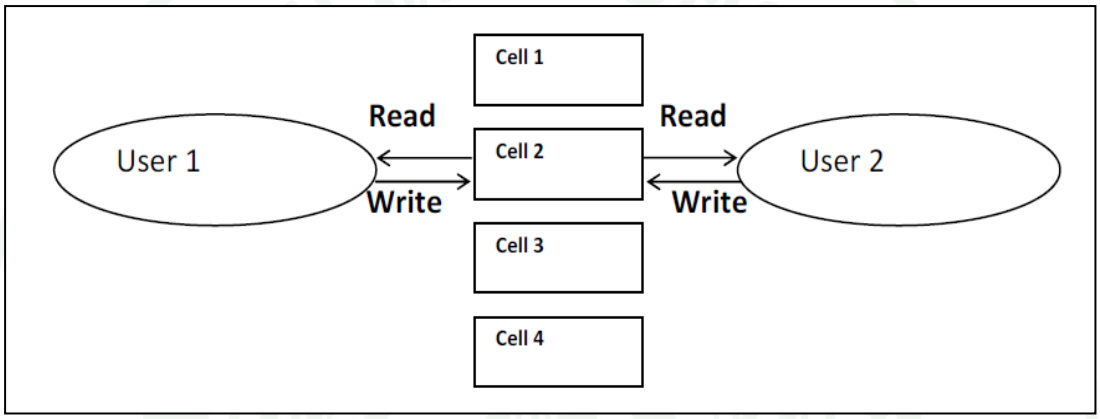
รูปแบบการเขียนข้อมูลที่เปลี่ยนแปลงตามฟังก์ชันการทำงานเต็ม (หรือแบบย่อ)

เรานิยามรูปแบบการเขียนข้อมูล คือการเปลี่ยนแปลงของข้อมูล และการนำข้อมูลนั้นไปใช้ต่อภายหลังการเปลี่ยนแปลง ภายหลังฟังก์ชันการทำงานแบบย่อถูกทำงาน การนำข้อมูลที่มีการเปลี่ยนแปลงไปใช้ต่อนั้นจะถูกนิยามภายใต้ฟังก์ชันการทำงานเดียวกันเท่านั้น เช่น ข้อมูลดังกล่าวมีสิทธิ์ถูกผู้ใช้คนอื่นไปใช้ต่อภายใต้ฟังก์ชันการทำงานเดิม เป็นต้น และฟังก์ชันการทำงานแบบย่อหนึ่งสามารถมีรูปแบบการเขียนข้อมูลได้มากกว่า 1 รูปแบบ เราได้นิยามรูปแบบการเขียนข้อมูลออกเป็น 3 รูปแบบ คือ ข้อมูลที่ถูกเขียนหลายครั้ง, ข้อมูลที่ถูกเขียนครั้งเดียว, และข้อมูลที่ถูกเขียนอย่างเดียว

นอกจากนี้เรายังได้นิยาม เซลล์ (Cell) คือ พื้นที่ที่ใช้ในการเก็บข้อมูล ภายในตารางข้อมูลเดียวกันนั้น แต่ละเซลล์จะมีตำแหน่งที่แตกต่างกัน งานวิจัยนี้จะนำเสนอผลกระทบจากการปรับฟังก์ชันการทำงานเต็มให้เป็นฟังก์ชันการทำงานแบบย่อ โดยวิธีการ: (1) แปลงจากข้อมูลที่เขียนครั้งเดียวให้กลายเป็น ข้อมูลที่ถูกเขียนอย่างเดียว, (2) ลบข้อมูลประเภทอื่นทิ้ง คงเหลือไว้แต่ข้อมูลประเภทข้อมูลที่ถูกเขียนอย่างเดียวเท่านั้น และ (3) ลบข้อมูลที่ถูกเขียนอย่างเดียว ซึ่งเกินความจำเป็นออกไป รูปแบบการเขียนข้อมูลแต่ละประเภทยังมีรูปแบบวิธีการสำเนาและ/หรือการกู้คืนข้อมูลที่แตกต่างกัน ระหว่างกลุ่มเมฆสาธารณะและกลุ่มเมฆปิดส่วนตนที่แตกต่างกันอีกด้วย

1. ข้อมูลที่ถูกเขียนหลายครั้ง (Multiple-write data)

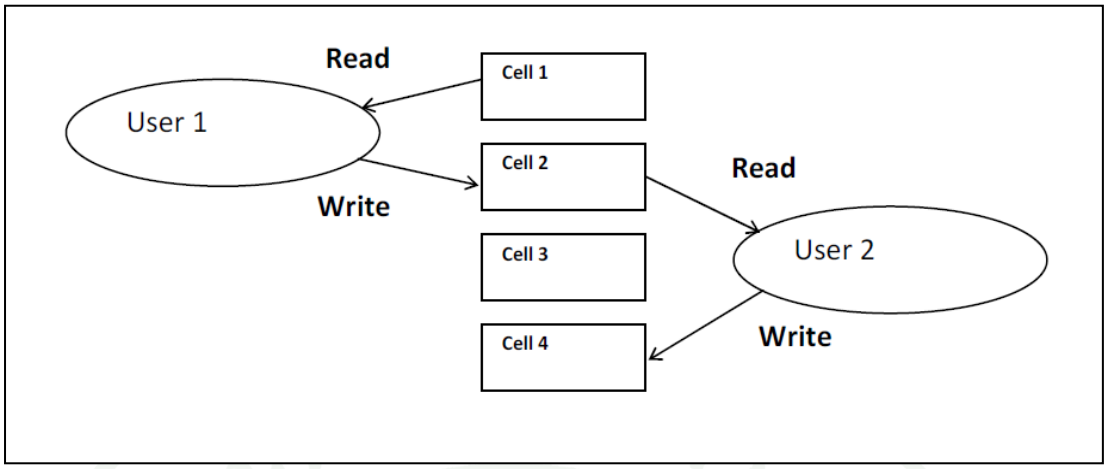
ตามภาพที่ 6 ลักษณะการเปลี่ยนแปลงของข้อมูลของประเภทนี้โดยผู้ใช้ คือ (1) อ่านค่าเดิมของข้อมูลเพื่อนำมาคำนวณและเขียนข้อมูลค่าใหม่ลงไปที่เซลล์เดิม, (2) อาจต้องมีการอ่านค่าจากเซลล์อื่นเพื่อมาคำนวณผลลัพธ์ด้วย, และ (3) เมื่อข้อมูลในเซลล์นั้นมีการเปลี่ยนแปลงแล้ว (ถูกเขียนแล้ว) ในเวลาต่อมา ข้อมูลนั้นอาจถูกนำไปใช้ในการอ่านหรือเขียนค่าใหม่ในอนาคต



ภาพที่ 6 ลักษณะการเปลี่ยนแปลงและการนำไปใช้ต่อของข้อมูลที่ถูกเขียนหลายครั้ง

2. ข้อมูลที่ถูกเขียนครั้งเดียว (Write-once data)

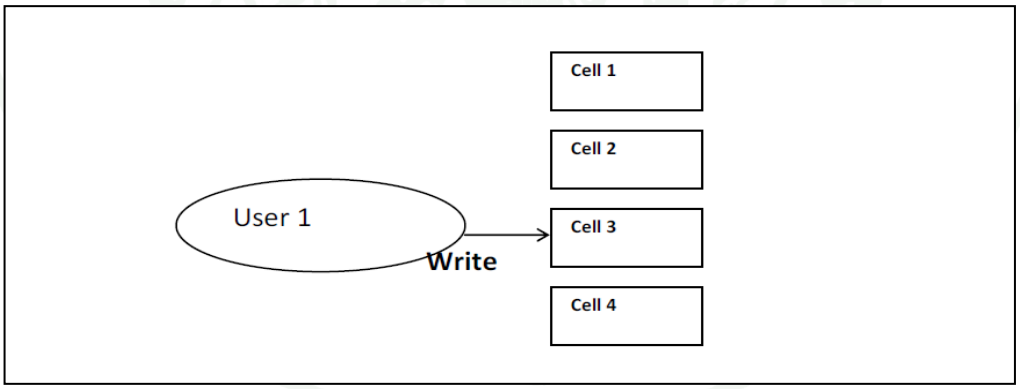
ตามภาพที่ 7 ลักษณะการเปลี่ยนแปลงของข้อมูลประเภทนี้โดยผู้ใช้ คือ (1) ก่อนที่จะมีการเปลี่ยนแปลง (การเขียน) ข้อมูลลงในเซลล์ใด จะต้องมีการตรวจสอบ (การอ่าน) ค่าข้อมูลที่เซลล์อื่นเพื่อนำมาคำนวณ และ (2) เมื่อทำการเปลี่ยนแปลงข้อมูลแล้ว ข้อมูลในเซลล์นั้นสามารถนำมาใช้อ่านจากผู้ใช้คนอื่นได้ในอนาคต



ภาพที่ 7 ลักษณะการเปลี่ยนแปลงและการนำไปใช้ต่อของข้อมูลที่ถูกเขียนครั้งเดียว

3. ข้อมูลที่ถูกเขียนอย่างเดียว (Write-only data)

ตามภาพที่ 8 ลักษณะการเปลี่ยนแปลงของข้อมูลประเภทนี้โดยผู้ใช้ คือ (1) ก่อนที่จะทำการเปลี่ยนแปลง (เขียน) ข้อมูลลงในเซลล์ใด ไม่จำเป็นจะต้องตรวจสอบ (อ่าน) ค่าข้อมูลใดๆ และ (2) เมื่อข้อมูลมีการเปลี่ยนแปลง (เขียน) แล้ว จะไม่มีการนำไปใช้ต่อในอนาคต



ภาพที่ 8 ลักษณะการเปลี่ยนแปลงและการนำไปใช้ต่อของข้อมูลที่ถูกเขียนอย่างเดียว

การจัดการข้อมูลทั้ง 3 รูปแบบ ในสภาวะต่างๆ

ระบบการทำงานสำรองจะมี 3 สภาวะ คือ สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ, สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน, และสภาวะคืนข้อมูลกลับสู่กลุ่มเมฆสาธารณะ

1. สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ

ถือว่าเป็นสภาวะที่ระบบจะทำงานตามปกติ สำหรับในเรื่องของรูปแบบการเขียนข้อมูล: ข้อมูลที่ถูกเขียนหลายครั้ง และข้อมูลที่ถูกเขียนครั้งเดียว จำเป็นจะต้องมีการอ่านข้อมูลใดๆ ก่อนทำการเปลี่ยนแปลง ดังนั้น จึงจำเป็นจะต้องทำการสำเนาข้อมูลที่ต้องใช้ในการอ่านไปยังกลุ่มเมฆปิดส่วนตนด้วย (1) เมื่อข้อมูลที่ถูกเขียนหลายครั้ง มีการเปลี่ยนแปลงบนกลุ่มเมฆสาธารณะ จะต้องมีการส่งค่าที่เปลี่ยนแปลงไปยังกลุ่มเมฆปิดส่วนตนด้วยทุกครั้ง เพราะค่าที่เปลี่ยนแปลงจะต้องนำไปใช้ในการอ่านและเขียนต่อในอนาคต, (2) ข้อมูลที่ถูกเขียนครั้งเดียว จะต้องมีการสำเนากลุ่มข้อมูลที่ถูกเขียนครั้งเดียวไปยังกลุ่มเมฆปิดส่วนตน ทุกๆหน่วยเวลา โดยไม่จำเป็นจะต้องสำเนาทุกครั้งที่มีการเปลี่ยนแปลง เพราะมีการนำไปใช้ในการอ่านอย่างเดียว, และสำหรับ (3) ข้อมูลที่ถูกเขียนอย่างเดียว ไม่จำเป็นต้องมีการสำเนาและ/หรือส่งข้อมูลใดๆไปเก็บยังกลุ่มเมฆปิดส่วนตน เพราะไม่จำเป็นต้องมีการอ่านข้อมูลใดๆ ก่อนข้อมูลประเภทนี้มีการเปลี่ยนแปลง ภายหลังจากข้อมูลนี้มีการเปลี่ยนแปลงแล้ว ไม่มีการนำไปใช้ต่อในอนาคต โดยสรุปที่สภาวะที่ใช้งานบนกลุ่มเมฆสาธารณะ ข้อมูลที่ถูกเขียนอย่างเดียว (1) ใช้เวลาในการเปลี่ยนแปลงน้อยกว่า, (2) ใช้ทรัพยากรในการประมวลผลน้อยกว่า ข้อมูลที่ถูกเขียนหลายครั้ง และข้อมูลที่ถูกเขียนครั้งเดียว

2. สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน

เมื่อระบบไม่สามารถใช้งานบนกลุ่มเมฆสาธารณะได้ ก็จะสลับมาทำงานที่ระบบการทำงานสำรองบนกลุ่มเมฆปิดส่วนตน เรียกว่า สภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน และเนื่องจากปริมาณทรัพยากรบนกลุ่มเมฆส่วนตนน้อยกว่าบนกลุ่มเมฆสาธารณะ ดังนั้นจึงจำเป็นจะต้องฟังก์ชันการทำงานเต็มบนกลุ่มเมฆสาธารณะให้เป็นฟังก์ชันการทำงานแบบย่อที่เหมาะสมกับกลุ่มเมฆปิดส่วนตน สำหรับในเรื่องของรูปแบบการเขียนข้อมูล: (1) ข้อมูลที่ถูกเขียนหลายครั้ง จะต้องมีการอ่านค่าเดิม และอ่านข้อมูลจากที่อื่นเพื่อทำการคำนวณ ก่อนทำการเปลี่ยนแปลง เมื่อเปลี่ยนแปลงแล้วมีการนำไปใช้ในการเขียน/อ่านต่อในอนาคต, (2) ข้อมูลที่ถูกเขียนครั้งเดียว จะต้องมีการอ่านข้อมูลจากที่อื่นเพื่อทำการคำนวณ ก่อนทำการเปลี่ยนแปลง เมื่อเปลี่ยนแปลงแล้วมีการ

นำไปใช้ในการอ่านอย่างเดียวในอนาคต, และ (3) ข้อมูลที่ถูกเขียนอย่างเดียว จะไม่มีการอ่านค่าใดๆ ก่อนทำการเปลี่ยนแปลง และเมื่อเปลี่ยนแปลงแล้วไม่มีการนำไปใช้ต่อในอนาคต โดยสรุปที่ใช้งานบนกลุ่มเมฆปิดส่วนตน ข้อมูลที่ถูกเขียนอย่างเดียว (1) ใช้เวลาในการเปลี่ยนแปลงน้อยกว่า, (2) ใช้ทรัพยากรในการประมวลผลน้อยกว่า ข้อมูลที่ถูกเขียนหลายครั้ง และข้อมูลที่ถูกเขียนครั้งเดียว

3. สภาวะคืนข้อมูลกลับสู่กลุ่มเมฆสาธารณะ

เมื่อกลุ่มเมฆสาธารณะสามารถกลับมาใช้งานได้ตามปกติ ก็จะมีการส่งค่าข้อมูลที่มีการเปลี่ยนแปลงภายในกลุ่มเมฆปิดส่วนตนทั้งหมดกลับไปยังกลุ่มเมฆสาธารณะ เรียกว่า สภาวะคืนข้อมูลกลับสู่กลุ่มเมฆสาธารณะ สำหรับในเรื่องของรูปแบบการเขียนข้อมูล: (1) ข้อมูลที่ถูกเขียนหลายครั้ง จะมีการส่งค่าที่เปลี่ยนแปลงกลับไปยังกลุ่มเมฆสาธารณะโดยทันที โดยไม่จำเป็นต้องมีการตรวจสอบใดๆ เพราะมีการอ่านก่อนเขียนไปแล้วในสภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน, (2) ข้อมูลที่ถูกเขียนครั้งเดียว จะมีการเพิ่มข้อมูลไปยังกลุ่มเมฆสาธารณะโดยทันทีเช่นเดียวกัน โดยไม่จำเป็นต้องมีการตรวจสอบใดๆ เพราะมีการอ่านก่อนเขียนไปแล้วในสภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน, และ (3) ข้อมูลที่ถูกเขียนอย่างเดียว จะมีการตรวจสอบทุกครั้งก่อนจะทำการเพิ่มข้อมูลทุกครั้งว่าสามารถเพิ่มข้อมูลได้หรือไม่ เพราะยังไม่มีการอ่านก่อนเขียนในสภาวะที่ใช้งานบนกลุ่มเมฆปิดส่วนตน โดยสรุปที่ใช้งานบนกลุ่มเมฆปิดส่วนตน ข้อมูลที่ถูกเขียนอย่างเดียว (1) ใช้เวลาในการเปลี่ยนแปลงมากกว่า, (2) ใช้ทรัพยากรในการประมวลผลมากกว่า ข้อมูลที่ถูกเขียนหลายครั้ง และข้อมูลที่ถูกเขียนครั้งเดียว

การปรับฟังก์ชันการทำงานเต็มเป็นฟังก์ชันการทำงานแบบย่อ

เราสามารถปรับฟังก์ชันการทำงานเต็มให้เป็นฟังก์ชันการทำงานแบบย่อโดย (1) แปลงจากข้อมูลที่เขียนครั้งเดียวให้กลายเป็น ข้อมูลที่ถูกเขียนอย่างเดียว (2) ลบข้อมูลประเภทอื่นทิ้ง ให้เหลือไว้แต่ข้อมูลประเภทข้อมูลที่ถูกเขียนอย่างเดียว เท่านั้น และ (3) ลบข้อมูลที่ถูกเขียนอย่างเดียวซึ่งเกินความจำเป็นออกไป

1. แปลงจากข้อมูลที่ถูกเขียนครั้งเดียวให้เป็นข้อมูลที่ถูกเขียนอย่างเดียว

ข้อมูลที่ถูกเขียนครั้งเดียวสามารถแปลงให้เป็นข้อมูลที่ถูกเขียนอย่างเดียวได้ เนื่องจากข้อมูลที่ถูกเขียนครั้งเดียว และข้อมูลที่ถูกเขียนอย่างเดียวนั้น ที่ตำแหน่งเซลล์ที่เคยมีการเขียนข้อมูลแล้ว จะไม่มีการเขียนค่าข้อมูลซ้ำ ส่วนเหตุผลที่ต้องแปลงจากข้อมูลที่ถูกเขียนครั้งเดียวให้เป็นข้อมูลที่ถูกเขียนอย่างเดียว เพราะก่อนข้อมูลที่ถูกเขียนครั้งเดียว มีการเปลี่ยนแปลงจะมีการอ่านข้อมูลที่ตำแหน่งอื่นก่อนทำการเปลี่ยนแปลง สำหรับข้อมูลที่ถูกเขียนอย่างเดียว เมื่อมีการเปลี่ยนแปลงจะไม่มีการอ่านข้อมูลตำแหน่งใดๆก่อนทำการเปลี่ยนแปลง จึงทำให้ข้อมูลที่ถูกเขียนอย่างเดียว ใช้เวลาในการประมวลผลน้อยกว่าข้อมูลที่ถูกเขียนครั้งเดียว แต่ถ้าข้อมูลที่ถูกเขียนครั้งเดียวใดๆมีการนำไปใช้ในการอ้างอิงจากข้อมูลที่ถูกเขียนครั้งเดียวประเภทอื่น จะไม่สามารถแปลงเป็นข้อมูลที่ถูกเขียนอย่างเดียวได้

2. การลบรูปแบบการเขียนข้อมูลประเภทอื่นยกเว้นข้อมูลที่ถูกเขียนอย่างเดียว

เนื่องจากก่อนที่ข้อมูลที่ถูกเขียนหลายครั้ง หรือข้อมูลที่ถูกเขียนครั้งเดียวนั้น มีการเปลี่ยนแปลงจะต้องมีการอ่านค่าข้อมูลจากเซลล์อื่นๆหรือค่าเดิมในเซลล์นั้นก่อน ในขณะที่ข้อมูลที่ถูกเขียนอย่างเดียว มีการเปลี่ยนแปลงจะไม่มีการอ่านข้อมูลตำแหน่งใดๆก่อนทำการเปลี่ยนแปลง จึงทำให้ข้อมูลที่ถูกเขียนอย่างเดียว ใช้เวลาในการประมวลผลน้อยกว่าข้อมูลที่ถูกเขียนครั้งเดียวและข้อมูลที่ถูกเขียนหลายครั้ง แต่ถ้าข้อมูลที่ถูกเขียนครั้งเดียวใดๆมีการนำไปใช้ในการอ้างอิงจากข้อมูลที่ถูกเขียนครั้งเดียวประเภทอื่น จะไม่สามารถถูกลบได้ สำหรับกรณีข้อมูลที่ถูกเขียนหลายครั้ง เนื่องจากข้อมูลที่ถูกเขียนหลายครั้ง เป็นข้อมูลที่ผู้ใช้ทำการเขียนร่วมกัน ซึ่งจะใช้ตำแหน่งเดิมในการเขียนข้อมูลเสมอและมีการเขียนซ้ำอย่างต่อเนื่อง ในเรื่องของการอ่านข้อมูลมันจะอ่านค่าที่เปลี่ยนแปลงล่าสุดเพียงค่าเดียวเท่านั้น ซึ่งถ้าเราแปลงให้เป็นรูปแบบการเขียนข้อมูลอีก 2 รูปแบบมันก็จะเก็บค่าใหม่ทุกครั้งที่ถูกเปลี่ยนแปลง (ถูกเขียน) ซึ่งจะเปลืองเนื้อที่ ถึงแม้จะแปลงข้อมูลชนิด

นี้ให้เป็นข้อมูลที่ถูกเขียนครั้งเดียวก็ตาม นอกจากจะเปลืองเนื้อที่ในการเก็บทุกค่าที่มีการเปลี่ยนแปลง (การเขียน) และยังจะจะเสียเวลาในการค้นหาข้อมูลที่เปลี่ยนแปลงล่าสุดจากข้อมูลที่มีการเปลี่ยนแปลงทั้งหมดที่เก็บ ดังนั้นการแปลงจาก ข้อมูลที่ถูกเขียนหลายครั้ง เป็นรูปแบบการเขียนข้อมูลชนิดใดๆ จึงเป็นวิธีที่ไม่เหมาะสม วิธีเดียวที่สามารถทำได้ คือ การลบข้อมูลที่ถูกเขียนหลายครั้ง ออกจากฟังก์ชันการทำงาน

3. การลบข้อมูลที่ถูกเขียนอย่างเดียวซึ่งเกินความจำเป็นออกไป

เราสามารถลบข้อมูลที่ถูกเขียนอย่างเดียวตัวใดๆทิ้งได้ แต่ภายหลังจากการลบนั้นฟังก์ชั่นการทำงานจะต้องยังคงสามารถใช้การได้

กรณีศึกษาระบบการจองตั๋วภาพยนตร์

ฟังก์ชันการทำงานเต็มของระบบการจองตั๋วภาพยนตร์ คือ (1) ตรวจสอบว่ามีจำนวนที่นั่งที่ยังว่างหรือไม่, (2) ตรวจสอบว่าหมายเลขที่นั่งที่เราต้องการนั้นถูกจองแล้วหรือยัง, (3) ทำการจองตั๋วพร้อมกับหมายเลขที่นั่ง, และ (4) ลดจำนวนที่นั่งที่ยังว่างไป 1 ที่นั่ง ประกอบด้วย รูปแบบการเขียน 5 ตัว คือ รหัสผู้ใช้ (Username), รหัสภาพยนตร์ (Movie ID), รอบที่ฉาย (Session), หมายเลขที่นั่ง (Seat ID), และจำนวนที่นั่งที่ยังว่าง (Vacant Seat)

รหัสผู้ใช้ เป็นข้อมูลที่ถูกเขียนอย่างเดียว เพราะเมื่อรหัสสมาชิกมีการเปลี่ยนแปลง (ถูกเขียน) โดยถูกเพิ่มเข้าไปเก็บในตารางข้อมูลการจองตั๋วภาพยนตร์ และจะไม่มีการนำมาใช้ในต่อในฟังก์ชันการทำงานของการจองตั๋วภาพยนตร์ที่ถูกใช้โดยผู้ใช้คนอื่น

รหัสภาพยนตร์ เป็นข้อมูลที่ถูกเขียนครั้งเดียว เพราะรหัสภาพยนตร์ เป็นข้อมูลที่ใช้เป็นตัวอ้างอิงในระดับแรก กล่าวคือ เมื่อรหัสภาพยนตร์มีการเปลี่ยนแปลง (ถูกเขียน) โดยถูกเพิ่มเข้าไปเก็บในตารางข้อมูลการจองตั๋วภาพยนตร์ จะถูกนำมาใช้ในการอ้างอิงจากรอบที่ฉาย (ถูกอ่านจากเซลล์อื่น) ว่าเป็นรอบที่ฉายของรหัสภาพยนตร์เรื่องใด

รอบที่ฉาย เป็นข้อมูลที่ถูกเขียนครั้งเดียว เพราะรอบที่ฉาย เป็นข้อมูลที่ใช้เป็นตัวอ้างอิง กล่าวคือ เมื่อรอบที่ฉายมีการเปลี่ยนแปลง (ถูกเขียน) โดยถูกเพิ่มเข้าไปเก็บในตารางข้อมูลการจอง

ตั๋วภาพยนตร์ จะถูกนำมาใช้ในการอ้างอิงจากหมายเลขที่นั่ง (ถูกอ่านจากเซลล์อื่น) ว่าเป็นหมายเลขที่นั่งของรอบที่ฉายใด

หมายเลขที่นั่ง เป็นข้อมูลที่ถูกเขียนครั้งเดียว เพราะจะมีการตรวจสอบก่อนว่าหมายเลขที่นั่งที่ผู้ใช้ต้องการนั้นถูกจองแล้วหรือยัง (เป็นการอ่านข้อมูลจากเซลล์อื่น) ถ้ายังไม่ถูกจองก็จะทำการจอง (เขียน) หมายเลขที่นั่งนั้น ภายหลังที่ผู้ใช้จองหมายเลขที่นั่งที่ต้องการแล้ว ผู้ใช้คนอื่นสามารถเข้ามาตรวจสอบ (อ่าน) หมายเลขที่นั่งที่เราได้ทำการจองไว้แล้วได้เช่นกัน

จำนวนที่นั่งที่ยังว่าง เป็นข้อมูลที่ถูกเขียนหลายครั้ง เพราะจะมีการตรวจสอบ (ถูกอ่าน) ว่ามีที่ว่างเหลืออยู่หรือไม่ ถ้ามีก็จะทำการเปลี่ยนแปลง (เขียนที่ตำแหน่งเซลล์เดิม) ข้อมูลนั้นโดยการเพิ่มจากค่าเดิมไปอีก 1 ค่า ภายหลังจากข้อมูลมีการเปลี่ยนแปลง อาจะมีผู้ใช้คนอื่นมาอ่านและเขียนข้อมูลค่าใหม่ได้ในอนาคตจำนวนที่นั่งที่ยังว่างของฟังก์ชันการทำงานของการจองตั๋วภาพยนตร์เป็นข้อมูลที่ถูกเขียนหลายครั้ง เพราะจะมีการตรวจสอบ (ถูกอ่าน) ว่ามีที่ว่างเหลืออยู่หรือไม่ ถ้ามีก็จะทำการเปลี่ยนแปลง (เขียนที่ตำแหน่งเซลล์เดิม) ข้อมูลนั้นโดยการเพิ่มจากค่าเดิมไปอีก 1 ค่า ภายหลังจากข้อมูลมีการเปลี่ยนแปลง อาจะมีผู้ใช้คนอื่นมาอ่านและเขียนข้อมูลค่าใหม่ได้ในอนาคต

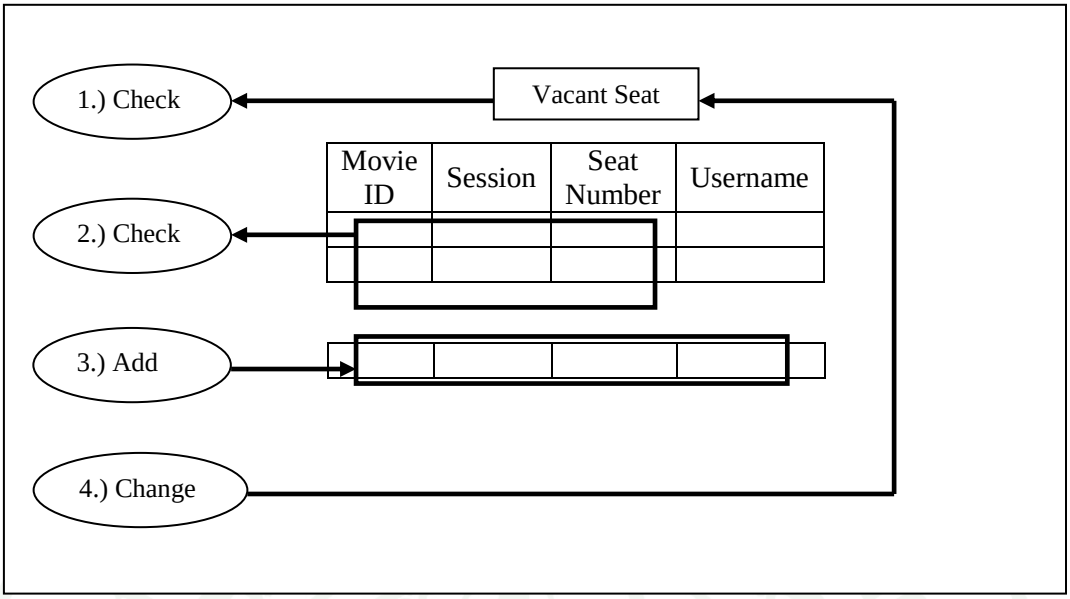
1. ปรับฟังก์ชันการทำงานเต็มของการจองตั๋วภาพยนตร์เป็นฟังก์ชันการทำงานแบบย่อ

เราได้ปรับจากฟังก์ชันการทำงานเต็มให้เป็นฟังก์ชันการทำงานแบบย่อ โดยให้มีการทำงานน้อยลง ทำให้จำนวนรูปแบบการเขียนข้อมูลที่เหลืออยู่นี้น้อยลงด้วย ซึ่งจะส่งผลกระทบให้คุณภาพในการให้บริการต่อผู้ใช้น้อยลง แต่สามารถรองรับจำนวนผู้ใช้ได้มากยิ่งขึ้น

2. ตัวอย่างการออกแบบฟังก์ชันการทำงานแบบย่อของระบบการจองตั๋วภาพยนตร์

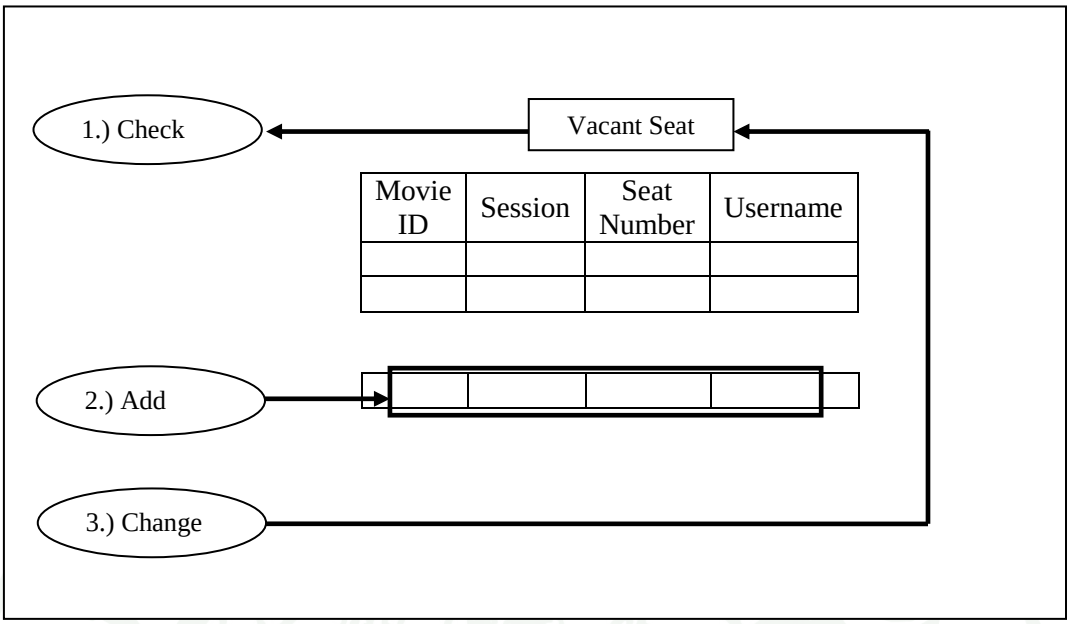
เราจะออกแบบระบบการทำงานสำรองบนกลุ่มเมฆปิดส่วนตน เพื่อใช้ทดแทนกลุ่มเมฆสาธารณะ จากที่ได้กล่าวไว้แล้วข้างต้นว่า กลุ่มเมฆปิดส่วนตนมีปริมาณทรัพยากรที่น้อยกว่ากลุ่มเมฆสาธารณะ เราจึงเลือกใช้วิธีปรับจากฟังก์ชันการทำงานเต็มบนกลุ่มเมฆสาธารณะให้เป็นฟังก์ชันการทำงานแบบย่อบนกลุ่มเมฆปิดส่วนตน โดยการ (1) แปลงข้อมูลที่ถูกเขียนครั้งเดียวให้เป็นข้อมูลที่ถูกเขียนอย่างเดียว, (2) ลบรูปแบบการเขียนข้อมูลประเภทอื่นยกเว้นข้อมูลที่ถูกเขียนอย่างเดียว, และ (3) การลบข้อมูลที่ถูกเขียนอย่างเดียวซึ่งเกินความจำเป็นออกไป เราได้ออกแบบระบบการทำงานสำรองตามการปรับมาเป็นฟังก์ชันการทำงานแบบย่อไว้ 3 รูปแบบ ดังนี้

2.1 ฟังก์ชันการทำงานแบบย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 1 ซึ่งเหมือนกันกับฟังก์ชันการทำงานเต็มของระบบการจองตั๋วภาพยนตร์ ประกอบด้วยขั้นตอนการทำงานตามภาพที่ 9 ดังนี้ (1) ตรวจสอบว่า มีจำนวนที่นั่งที่ยังว่างของรอบฉายหนังนั้นหรือไม่ โดยอ่านข้อมูลจำนวนที่นั่งเหลือจากเซลล์ข้อมูลชื่อ Vacant Seat ถ้ายังมีที่นั่งว่างจะ (2) ตรวจสอบว่าหมายเลขที่นั่งที่ผู้จองต้องการนั้นถูกจองแล้วหรือยัง โดยอ่านข้อมูลที่นั่งที่ถูกจองไปแล้วทั้งหมดในรอบฉายของหนังที่ผู้ใช้เลือก เซลล์ข้อมูลที่ต้องอ่านคือ Seat Number, Session, MovieID ซึ่งเก็บเบอร์ที่นั่ง รอบฉาย และรหัสหนัง ตามลำดับ ถ้าที่นั่งที่ผู้ใช้ต้องการยังไม่ถูกจอง ระบบจะ (3) ทำการจองตั๋วโดยบันทึกชื่อผู้ใช้ (username) พร้อมกับหมายเลขที่นั่ง (Seat Number) รอบฉาย (Session) และรหัสหนัง (MovieID) ที่ผู้ใช้เลือกลงไปในเซลล์กลุ่มใหม่ ข้อมูลกลุ่มนี้ (ยกเว้น username) จะถูกอ่านเมื่อมีผู้ใช้คนต่อไปเข้ามาจองตั๋ว สุดท้ายระบบจะ (4) ลดจำนวนที่นั่งที่ยังว่าง (Vacant Seat) ไป 1 ที่นั่ง จากขั้นตอนการทำงานนี้จะเห็นว่าข้อมูลจำนวนที่นั่งว่าง (Vacant Seat) จะเป็นข้อมูลประเภทเขียนหลายครั้ง เพราะในการจองแต่ละครั้งจะมีการอ่านและเขียนข้อมูลนี้ตลอดจนกว่าที่นั่งจะเต็ม ส่วนข้อมูลการจองของแต่ละคนนั้นประกอบไปด้วยข้อมูลเขียนครั้งเดียวคือเลขที่นั่ง รอบฉาย และรหัสหนัง และข้อมูลเขียนอย่างเดียวคือชื่อผู้ใช้ สาเหตุที่เลขที่นั่ง รอบฉาย และรหัสหนังเป็นข้อมูลเขียนครั้งเดียวเพราะเมื่อมีการบันทึกการจองจากผู้ใช้คนหนึ่งไปแล้วข้อมูลนี้จะถูกอ่านโดยผู้ใช้คนต่อๆ ไปเพื่อตรวจสอบว่าเลขที่นั่งใดถูกจองไปแล้วบ้าง ส่วนชื่อผู้ใช้นั้นเป็นข้อมูลประเภทเขียนอย่างเดียวเพราะไม่จำเป็นต้องใช้ข้อมูลนี้ในการจองครั้งต่อไปของผู้ใช้คนอื่น



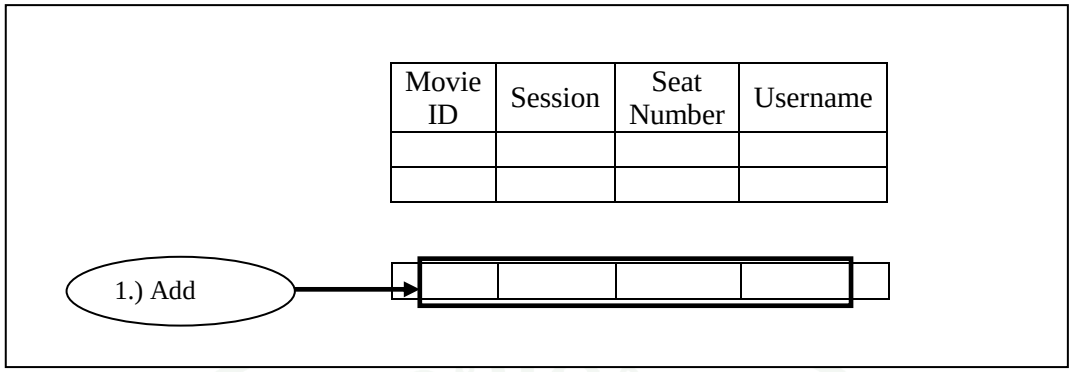
ภาพที่ 9 ฟังก์ชันการทำงานย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 1

2.2 ฟังก์ชันการทำงานแบบย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 2 นั้นจะตัดส่วนการตรวจสอบเลขที่นั่งที่ถูกจองไปแล้วออกไป ทำให้คุณภาพในการให้บริการลดลงเพราะไม่สามารถรับประกันได้ว่าผู้จองจะได้หมายเลขที่นั่งที่ตนเลือกอย่างแน่นอน (อาจมีผู้ใช้มากกว่าหนึ่งคนจองที่นั่งเดียวกันได้) ระบบรับประกันได้เพียงว่าจะมีที่นั่งเพียงพอสำหรับผู้ที่จองตั๋วทุกคนเท่านั้น ขั้นตอนการทำงานเป็นดังรูปที่ 10 ดังนี้ (1) ตรวจสอบว่ามีจำนวนที่นั่งที่ยังว่างของรอบฉายหนังนั้นหรือไม่ โดยอ่านข้อมูลจำนวนที่นั่งเหลือจากเซลล์ข้อมูลชื่อ Vacant Seat ถ้ายังมีที่นั่งว่างจะ (2) ทำการจองตั๋วโดยบันทึกชื่อผู้ใช้ (username) พร้อมกับหมายเลขที่นั่ง (Seat Number) รอบฉาย (Session) และรหัสหนัง (MovieID) ที่ผู้ใช้เลือกลงไปในเซลล์กลุ่มใหม่ และ (3) ลดจำนวนที่นั่งที่ยังว่าง (Vacant Seat) ไป 1 ที่นั่ง จากขั้นตอนการทำงานนี้จะเห็นว่าข้อมูลจำนวนที่นั่งว่าง (Vacant Seat) จะยังเป็นข้อมูลประเภทเขียนหลายครั้งและชื่อผู้ใช้จะยังเป็นข้อมูลประเภทเขียนอย่างเดียวเหมือนการทำงานแบบที่ 1 แต่หมายเลขที่นั่ง (Seat Number) รอบฉาย (Session) และรหัสหนัง (MovieID) จะเปลี่ยนจากข้อมูลเขียนครั้งเดียวไปเป็นข้อมูลเขียนอย่างเดียว เพราะระบบไม่มีการตรวจสอบข้อมูลชุดนี้ในการจองตั๋วแต่ละครั้งอีกต่อไป



ภาพที่ 10 ฟังก์ชันการทำงานย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 2

2.3 ฟังก์ชันการทำงานแบบย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 3 มีการทำงานดังนี้ คือ ไม่มีการตรวจสอบว่ามีจำนวนที่นั่งที่ยังว่างหรือไม่ และไม่มีการตรวจสอบว่าหมายเลขที่นั่งที่ตนจองนั้นถูกผู้ใช้คนอื่นจองไปก่อนหน้านี้แล้วหรือยัง แต่จะเป็นการรับฝากข้อมูลการจองตั๋วไว้กับระบบ และไม่มีการลดจำนวนที่นั่งที่ยังว่างลงไป 1 ที่นั่ง เมื่อเวลาผ่านไประบบจะทำการจองตั๋วให้เองโดยอัตโนมัติตามลำดับ ระบบจะไม่มีการรับประกันว่าผู้ใช้สามารถจองตั๋วได้ เพราะเสมือนเป็นเพียงการรับฝากเรื่องเท่านั้น ซึ่งฟังก์ชันการทำงานแบบย่อแบบที่ 3 จะมีคุณภาพในการให้บริการต่ำยิ่งกว่าฟังก์ชั่นการทำงานแบบย่อแบบที่ 2 โดยทำการปรับจากฟังก์ชั่นการทำงานย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 2 คือ ถ้าเราทำการลบข้อมูลจำนวนที่นั่งที่ยังว่าง จะทำให้ฟังก์ชั่นการทำงานแบบย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 3 มีรูปแบบการเขียนข้อมูลอยู่ 4 ตัว ดังภาพที่ 11 คือ (1) รหัสผู้ใช้ เป็นข้อมูลที่ถูกเขียนอย่างเดียว เพราะสามารถเปลี่ยนแปลง (ถูกเขียน) ได้เลย และภายหลังจากการเปลี่ยนแปลงไม่มีการนำไปใช้ต่อ, (2) รหัสภาพยนตร์ เป็นข้อมูลที่ถูกเขียนอย่างเดียว เพราะสามารถเปลี่ยนแปลง (ถูกเขียน) ได้เลย และภายหลังจากการเปลี่ยนแปลงไม่มีการนำไปใช้ต่อ, (3) รอบที่ฉาย เป็นข้อมูลที่ถูกเขียนอย่างเดียว เพราะสามารถเปลี่ยนแปลง (ถูกเขียน) ได้เลย และภายหลังจากการเปลี่ยนแปลงไม่มีการนำไปใช้ต่อ, (4) หมายเลขที่นั่ง เป็นข้อมูลที่ถูกเขียนอย่างเดียว เพราะสามารถเปลี่ยนแปลง (ถูกเขียน) ได้เลย และภายหลังจากการเปลี่ยนแปลงไม่มีการนำไปใช้ต่อ



ภาพที่ 11 ฟังก์ชันการทำงานย่อของระบบการจองตั๋วภาพยนตร์แบบที่ 3

ตารางที่ 1 เปรียบเทียบคุณภาพของระบบ

ฟังก์ชันการทำงานแบบย่อ	ข้อมูลที่ถูกเขียนหลายครั้ง	ข้อมูลที่ถูกเขียนครั้งเดียว	ข้อมูลที่ถูกเขียนอย่างเดียว	คุณภาพของระบบ
แบบที่ 1	1	3	1	เมื่อผู้ใช้จองตั๋วพร้อมหมายเลขที่นั่งสำเร็จ จะได้ที่นั่งหมายเลขนั้นแน่นอน
แบบที่ 2	1	0	4	เมื่อผู้ใช้จองตั๋วสำเร็จ จะได้ที่นั่งแน่นอน แต่ไม่รับประกันว่าจะได้หมายเลขที่นั่งที่ต้องการ
แบบที่ 3	0	0	4	เมื่อผู้ใช้จองตั๋วสำเร็จ ไม่มีการรับประกันว่าจะได้ที่นั่งแน่นอน

ผลและวิจารณ์

ผล

ผู้วิจัยได้ออกแบบฟังก์ชั่นการทำงานแบบย่อตามข้อ 6.2 ไว้ทั้ง 3 รูปแบบ และพัฒนาเป็นเว็บแอพพลิเคชัน โดยใช้ภาษา JSP และใช้ MySQL เป็นฐานข้อมูล ซึ่งจะทำการวัดประสิทธิภาพของฟังก์ชันการทำงานแบบย่อทั้ง 3 รูปแบบ โดยการทำ Load Testing ภายใต้สภาวะแวดล้อมเดียวกันทั้ง 3 รูปแบบ โดยใช้โปรแกรม JMeter แบบมีการเพิ่มผู้ใช้เองโดยอัตโนมัติ และระบบจะหยุดการทำงานเองเมื่อระบบไม่สามารถทำงานตามเงื่อนไขที่เราได้กำหนดไว้ รวมไปถึงทำการเปรียบเทียบระบบของฟังก์ชันการทำงานแบบย่อทั้ง 3 ระบบ ตามกระบวนการ 7 ขั้นตอนของการทดสอบประสิทธิภาพเว็บแอพพลิเคชั่น พร้อมทั้งวิเคราะห์ผลการทดลอง ดังนี้

กำหนดสภาวะแวดล้อมในการทดสอบ

กำหนดสภาวะแวดล้อมในการสภาวะแวดล้อมในการทดสอบประกอบไปด้วยคอมพิวเตอร์ 2 เครื่อง เชื่อมต่อกันผ่านทางสาย LAN เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเครื่องแม่ข่ายใช้มีคุณสมบัติคือซีพียู Core2Duo ความเร็ว 2.16 GHz และหน่วยความจำหลักขนาด 2 GB, เครื่องที่สองทำหน้าที่เป็นเครื่องผู้ใช้ มีคุณสมบัติ คือ ซีพียู AMD E-450 ความเร็ว 1.65 GHz และหน่วยความจำหลักขนาด 2 GB ซึ่งจะใช้โปรแกรม JMeter สร้างกลุ่มคำร้องขอส่งไปยังเครื่องแม่ข่าย และดูผลลัพธ์

กำหนดเงื่อนไขที่ยอมรับได้

กำหนดผลลัพธ์จะขัดกับเงื่อนไขที่ยอมรับได้ เมื่อ (1) ภายใน 1 วินาที เวลาในการตอบสนองต่อผู้ใช้มากกว่า 10000 มิลลิวินาที และ Throughput น้อยกว่า 5 หรือ (2) ภายใน 10 วินาที ใช้หน่วยประมวลผลเกินกว่า 70% และอัตราความผิดพลาดมากกว่า 50%

วางแผนและออกแบบการทดลอง

เราจำลองกลุ่มของผู้ใช้จากเครื่องผู้ใช้ส่งไปยังเครื่องแม่ข่าย โดยเพิ่มจำนวนผู้ใช้ไปเรื่อยๆ จนกระทั่งการประมวลผลของระบบขัดกับข้อกำหนดเงื่อนไขที่ยอมรับได้ในข้อ 2 เราทำการทดสอบทั้ง 3 ระบบ แต่ละระบบเราทำการทดสอบ 30 ครั้ง แล้วนำผลหาค่าเฉลี่ย เพื่อเปรียบเทียบประสิทธิภาพว่าระบบที่ใช้ฟังก์ชันการทำงานแบบย่อแบบใดสามารถรองรับจำนวนผู้ใช้ได้มากที่สุด

ติดตั้งสภาวะแวดล้อมในการทดสอบ

ในเครื่องแม่ข่าย เราติดตั้ง Apache TOMCAT 7.0.11 เป็นเว็บเซิร์ฟเวอร์เพื่อรองรับจำนวนผู้ใช้, ใช้ MySQL เป็นฐานข้อมูล ระบบปฏิบัติการที่ใช้บนเครื่องแม่ข่าย คือ Windows Vista Ultimate ในเครื่องผู้ใช้ เราจำลองกลุ่มผู้ใช้มาจากโปรแกรม JMeter add-ins และระบบปฏิบัติการที่ใช้คือ Windows 7

กำหนดค่าต่างๆ ตามการทดลองที่ได้ออกแบบไว้

อย่างแรก เรากำหนด Thread Group คือ จำนวนผู้ใช้กลุ่มแรกมี 100 คน และค่อยๆเพิ่มขึ้นทุกๆ 10 คน ในทุกๆ 5 วินาที เงื่อนไขที่ยอมรับได้ ที่ถูกกำหนดไว้ใน ข้อ 2 หลังจากนั้น เราก็กำหนด IP Address, port และตัวแปรสุ่มต่างๆ เพื่อส่งไปยังเครื่องแม่ข่าย และสุดท้ายเพิ่ม Aggregate Report เพื่อดูผลการทดลองว่าสามารถรองรับจำนวนผู้ใช้ได้มากสุดเท่าใด

ทำการทดลอง

ฟังก์ชันการทำงานแบบย่อทั้ง 3 รูปแบบจะถูกเข้าไปทดสอบ 30 ครั้ง ภายใต้สภาวะแวดล้อมที่ได้กำหนดไว้ และนำผลของแต่ละครั้งมาหาค่าเฉลี่ยเพื่อเปรียบเทียบว่าฟังก์ชันการทำงานแบบย่อแบบใดสามารถรองรับจำนวนผู้ใช้ได้มากที่สุด โดยไม่ขัดกับเงื่อนไขและการยอมรับที่กำหนดไว้ในข้อ 2

วิเคราะห์ผลการทดลอง

จากตาราง 1 จำนวนผู้ใช้ที่รองรับมากที่สุด (มาจากค่าเฉลี่ยจากการทำการทดลอง 30 ครั้ง) ของฟังก์ชันการทำงาหลักแบบที่ 1 คือ 469 คน ในขณะที่ฟังก์ชันการทำงานแบบย่อแบบที่ 2 คือ 572 คน และฟังก์ชันการทำงานแบบย่อแบบที่ 3 คือ 1755 คน (1) เมื่อเราปรับจากฟังก์ชันการทำงานแบบย่อแบบที่ 1 เป็นฟังก์ชันการทำงานแบบย่อแบบที่ 2 โดยการแปลงข้อมูลที่ถูกเขียนครั้งเดียวให้เป็นข้อมูลที่ถูกเขียนอย่างเดียว ทำให้ระบบรองรับผู้ใช้ได้เพิ่มขึ้น 103 คน มีประสิทธิภาพจะเพิ่มขึ้นถึง 21.96% (2) เมื่อเราปรับจากฟังก์ชันการทำงานแบบย่อแบบที่ 2 เป็นฟังก์ชันการทำงานแบบย่อแบบที่ 3 โดยการลบข้อมูลที่ถูกเขียนหลายครั้ง ทำให้ระบบรองรับผู้ใช้ได้เพิ่มขึ้น 1183 คน ประสิทธิภาพจะเพิ่มขึ้น 206.81% ดังนั้น ข้อมูลที่ถูกเขียนหลายครั้งมีปริมาณภาระงานที่มากที่สุด ข้อมูลที่ถูกเขียนครั้งเดียวมีปริมาณภาระงานรองลงมา และข้อมูลที่ถูกเขียนอย่างเดียวมีปริมาณภาระงานที่น้อยที่สุด

ตารางที่ 2 สรุปผลการทดลอง 3 รูปแบบ

ฟังก์ชันการทำงานแบบย่อ	MAX	MIN	AVERAGE	SD
แบบที่ 1	571	293	469	74.84
แบบที่ 2	642	357	572	70.02
แบบที่ 3	2047	1441	1755	146.55

ตารางที่ 3 สรุปคุณภาพของระบบ

ฟังก์ชันการทำงานแบบย่อ	ข้อมูลที่ถูกเขียนหลายครั้ง	ข้อมูลที่ถูกเขียนครั้งเดียว	ข้อมูลที่ถูกเขียนอย่างเดียว	ความสามารถของระบบ
แบบที่ 1	1	3	1	- สามารถตรวจสอบจำนวนที่นั่งที่ยังว่างได้ - สามารถจองตั๋วพร้อมหมายเลขที่นั่งได้ - จำนวนที่นั่งที่ยังว่างจะลดลง 1 ตำแหน่ง เมื่อผู้ใช้ทำการจอง
แบบที่ 2	1	0	4	- สามารถตรวจสอบจำนวนที่นั่งที่ยังว่างได้ - สามารถจองตั๋วแต่ไม่สามารถการันตีว่าจองหมายเลขที่นั่งได้ - จำนวนที่นั่งที่ยังว่างจะลดลง 1 ตำแหน่ง เมื่อผู้ใช้ทำการจอง
แบบที่ 3	0	0	4	- สามารถจองตั๋วแต่ไม่สามารถการันตีว่าจองหมายเลขที่นั่งได้

วิจารณ์

จากตารางที่ 1 แสดงให้เห็นผลลัพธ์ของการทดสอบประสิทธิภาพ คือ เมื่อเปรียบเทียบฟังก์ชันการทำงานแบบย่อแบบที่ 1, ฟังก์ชันการทำงานแบบย่อแบบที่ 2, และฟังก์ชันการทำงานแบบย่อแบบที่ 3 (1) ฟังก์ชันการทำงานแบบย่อแบบที่ 1 ประกอบด้วย ข้อมูลที่ถูกเขียนอย่างเดียว 1 ตัว, มีข้อมูลที่ถูกเขียนครั้งเดียว 3 ตัว, และมีข้อมูลที่ถูกเขียนหลายครั้ง 1 ตัว, (2) ฟังก์ชันการทำงานแบบย่อแบบที่ 2 ประกอบด้วย ข้อมูลที่ถูกเขียนอย่างเดียว 4 ตัว, และมีข้อมูลที่ถูกเขียนหลายครั้ง 1 ตัว, และ (3) ฟังก์ชันการทำงานแบบย่อแบบที่ 3 ประกอบด้วย ข้อมูลที่ถูกเขียนอย่างเดียว 4 ตัว

เมื่อเราทำการปรับฟังก์ชันการทำงานแบบย่อแบบที่ 1 ให้เป็นฟังก์ชันการทำงานแบบย่อแบบที่ 2 จะทำให้ประสิทธิภาพเพิ่มขึ้น 21.96% เพราะฟังก์ชั่นการทำงานแบบย่อแบบที่ 2 ได้แปลงจากข้อมูลที่ถูกเขียนครั้งเดียวให้เป็นข้อมูลที่ถูกเขียนอย่างเดียว และข้อมูลที่ถูกเขียนครั้งเดียวก็จำเป็นจะต้องอ่านค่าข้อมูลจากเซลล์หนึ่ง เพื่อทำการคำนวณก่อนเขียนข้อมูลลงไปยังอีกเซลล์หนึ่ง แต่ข้อมูลที่ถูกเขียนอย่างเดียว ไม่จำเป็นต้องอ่านค่าข้อมูลใดๆ เพื่อคำนวณก่อนทำการเขียนข้อมูล

เมื่อเราทำการปรับฟังก์ชันการทำงานแบบย่อแบบที่ 2 ให้เป็นฟังก์ชันการทำงานแบบย่อแบบที่ 3 จะทำให้ประสิทธิภาพเพิ่มขึ้น 206.81% เพราะฟังก์ชั่นการทำงานแบบย่อแบบที่ 3 ได้ลบข้อมูลที่ถูกเขียนหลายครั้งออก เพราะข้อมูลที่ถูกเขียนหลายครั้ง จำเป็นจะต้องอ่านค่าข้อมูลก่อนหน้า เพื่อทำการคำนวณก่อนเขียนข้อมูล และเขียนลงไปยังตำแหน่งเซลล์เดิม

สรุปและข้อเสนอแนะ

สรุป

งานวิจัยนี้นำเสนอเทคนิคการออกแบบกลุ่มเมฆปิดส่วนตนเพื่อใช้เป็นระบบการทำงานสำรองทดแทนกลุ่มเมฆสาธารณะ แต่เนื่องจากกลุ่มเมฆปิดส่วนตนมีปริมาณทรัพยากรที่น้อยกว่ากลุ่มเมฆสาธารณะ จึงไม่สามารถรองรับปริมาณงานทั้งหมดได้เท่ากับกลุ่มเมฆสาธารณะ เราจึงแก้ปัญหาโดยการปรับฟังก์ชันการทำงานเต็ม เป็นฟังก์ชันการทำงานแบบย่อ โดยการลดคุณภาพในการให้บริการของระบบ เพื่อให้ระบบสามารถรองรับจำนวนผู้ใช้ได้มากยิ่งขึ้น

เราได้นิยามรูปแบบการเขียนข้อมูล 3 รูปแบบ คือ ข้อมูลที่ถูกเขียนหลายครั้ง, ข้อมูลที่ถูกเขียนครั้งเดียว, และข้อมูลที่ถูกเขียนอย่างเดียวและในการปรับจากฟังก์ชันการทำงานเต็มให้เป็นฟังก์ชันการทำงานแบบย่อสามารถทำได้ 3 วิธีคือ (1) การแปลงจากข้อมูลที่ถูกเขียนครั้งเดียวเป็นข้อมูลที่ถูกเขียนอย่างเดียว, (2) ลบรูปแบบการเขียนข้อมูลประเภทอื่นทิ้ง ให้เหลือไว้แต่ข้อมูลที่ถูกเขียนอย่างเดียว, และ (3) ลบข้อมูลที่ถูกเขียนอย่างเดียวที่เกินความจำเป็นออกไป

ผลการทดลองแสดงให้เห็นว่าการออกแบบกลุ่มเมฆปิดส่วนตน เพื่อใช้ทดแทนกลุ่มเมฆสาธารณะให้มีประสิทธิภาพสูงสุดนั้นจะต้องแปลง/ลบรูปแบบการเขียนข้อมูลประเภทอื่น และให้เหลือไว้แต่ข้อมูลที่ถูกเขียนอย่างเดียว แต่จะส่งผลให้คุณภาพในการบริการลดลง ผู้ออกแบบระบบจะต้องปรับค่าระหว่างคุณภาพในการให้บริการของระบบ และประสิทธิภาพของระบบอย่างเหมาะสม

ข้อเสนอแนะ

งานวิจัยนี้เป็นการปรับฟังก์ชันการทำงานเต็ม เป็นฟังก์ชันการทำงานแบบย่อโดยฟังก์ชันการทำงานที่ใช้ในการเพิ่ม และการเปลี่ยนแปลงข้อมูลเฉพาะในส่วนที่เกี่ยวข้องกับการเพิ่มข้อมูลเท่านั้น ยังไม่รวมถึงฟังก์ชันการทำงานที่ใช้ในการลบข้อมูล และ/หรือการเปลี่ยนแปลงข้อมูลเฉพาะในส่วนที่เกี่ยวข้องกับการลบข้อมูล ซึ่งสามารถเป็นประเด็นในงานวิจัยอื่นในอนาคตได้ ในเรื่องของรูปแบบการเขียนข้อมูล ที่เราได้แบ่งตาม ลักษณะในการเปลี่ยนแปลง และการนำข้อมูลไปใช้ต่อ มีประเด็นที่น่าสนใจคือ

1. วิธีการในการสำเนาข้อมูลที่แตกต่างกันระหว่างกลุ่มเมฆปิดส่วนตนและกลุ่มเมฆสาธารณะ, สำหรับรูปแบบการเขียนข้อมูลทั้ง 3 ประเภทนี้

2. อาจนำข้อมูลไปใช้ต่อของรูปแบบการเขียนข้อมูลทั้ง 3 ประเภทนี้ สามารถเป็นแนวทางในการแก้ปัญหาในด้านความปลอดภัยของการประมวลผลบนเซิร์ฟเวอร์กลุ่มเมฆ (Cloud Security) ได้, รวมไปถึง

3. อาจนำไปประยุกต์ใช้กับข้อมูลขนาดใหญ่ (Big Data) ได้

เอกสารและสิ่งอ้างอิง

Peery, C., T.D. Nguyen and F.M. Cuenca-Acuna. 2006. Removing the Availability Management Overheads of Federated Content Sharing Systems, pp. 5-17. *In* **25th IEEE Symposium on Reliable Distributed Systems (SRDS'06).** IEEE Computer Society, Leeds, UK.

Wei, B., C. Lin and X. Kong. 2011. Dependability Modeling and Analysis for the Virtual Data Center of Cloud Computing, pp. 784-789. *In* **2011 IEEE International Conference on High Performance Computing and Communications.** HPCC, Alberta, Canada

Yang, Q., W. Xiao and J. Ren. 2006. PRINS: Optimizing Performance of Reliable Internet Storages, pp. 32-36. *In* **26th IEEE International Conference on Distributed Computing Systems (ICDCS'06).** IEEE Computer Society, Lisboa, Portugal.

Wu, M., X-H. Sun and H. Jin. 2007. Performance under failures of high-end computing, pp. 221-229. *In* **Proc. of SC'07 2007**. ACM/IEEE Supercomputer, NV, USA.

Zorzo, A.F. and F.R. Meneguzzi. 2005. An agent model for fault-tolerant systems, pp. 60–65. *In* **Proceedings of the 2005 ACM symposium on Applied Computing**. ACM, New Mexico, USA.

Bora, S. 2005. A Fault Tolerant System Using Collaborative Agents. **Lecture Notes in Computer Science** 3949: 211-218.

Abdelzaher, T.F. and N. Bhatti. 1999. Web content adaptation to improve server overload behavior. **In The International Journal of Computer and Telecommunications Networking** 31: 11-16.

Gopshtein, M. and D.G. Feitelson. 2011. Trading off Quality for Throughput Using Content Adaptation in Web Servers, pp. 6-19. *In* **Proceedings of SYSTOR 2011**. 4th Annual Haifa Experimental Systems Conference, Haifa, Israel.

Schwarz, H., D. Marpe and T. Wiegand. 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, pp. 1103-1119. *In* **IEEE Transactions on Circuits and Systems for Video Technology**.

Mell, P. and T. Grance. 2011. **The NIST Definition of Cloud Computing**. Special Publication 800-145, USA.

Meier, J.D., C. Farre, P. Bansode, S. Barber and D. Rea. 2007. P**atterns & Practices: Performance Testing Guidance for Web Applications**. Microsoft Press, USA.

Hansen, H.K. 2003. **Load Testing your Applications with Apache JMeter**. Apache JMeter 1.8, Java Boutique, USA.

Vaquero, L.M., L. Rodero-Merino, J. Caceres and M. Linder. 2009. A Break in the Clouds: Towards a Cloud Definition, pp. 50 – 55. *In* **ACM SIGCOMM Computer Communication Review** 39: 1.

IDC eXchange. 2009. **New IDC IT Cloud Services Survey: Top Benefits and Challenges**. Available Source: http://blogs.idc.com/ie/?p=730, September 6, 2012.

InformationWeek**.** 2012. **Amazon Outage Hits Netflix, Heroku, Pinterest, Instagram**. Available Source: http://www.informationweek.com/cloud-computing/infrastructure/amazon-outage-hits-netflix-heroku-pinter/240003096, December 2, 2012.

Ramgovind, S., M.M. Eloff and E. Smith. 2010. The management of security in Cloud computing, pp. 1-7. *In* **Information Security for South Asia (ISSA**).

Baburajan, R. 2011. **The Rising Cloud Storage Market Opportunity Strengthens Vendors**. infoTECH, USA.

ภาคผนวก

ภาคผนวก ก

ผลการทดลอง

ผลการทดลองของระบบ

ทำการทดลองทั้ง 3 รูปแบบฟังก์ชันการทำงานแบบย่อ รูปแบบละ 30 ครั้ง เพื่อวัดจำนวนผู้ใช้ที่ระบบสามารถรองรับได้ โดยไม่ขัดกับเงื่อนไขตั้งไว้ นำผลการทดลองทั้ง 30 ครั้ง ของแต่ละรูปแบบมาหาค่าเฉลี่ย แล้วทำการเปรียบเทียบจำนวนผู้ใช้ที่สามารถรองรับได้ทั้ง 3 รูปแบบ

ตารางผนวกที่ ก1 ผลการทดลอง 30 ครั้งของฟังก์ชันการทำงานแบบย่อแบบที่ 1

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
1	293	22579	21728	37495	1040	41997	0	2.321859	1.049871
2	527	10711	10969	16625	1071	32127	0	4.678248	2.115419
3	310	18733	20028	25296	3362	30285	0	2.73135	1.235182
4	304	19696	21469	25900	2148	27061	0	2.554429	1.154981
5	534	10202	9765	19083	735	25936	0	4.9292	2.228893
6	482	11488	11939	18944	926	22828	0	4.39184	1.985817
7	523	10665	10101	18011	954	32785	0	4.735732	2.141413
8	571	9880	9506	16706	1037	26175	0	6.148272	2.327889
9	446	12263	11833	20650	1230	35087	0	4.105264	1.856566

ตารางผนวกที่ ก1 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
10	562	9813	9355	15511	860	29587	0	6.137018	2.322909
11	532	10580	9857	17667	802	28791	0	4.763482	2.154221
12	431	13108	15696	19603	1364	31649	0	3.840739	1.736688
13	552	10040	9970	16956	325	32295	0	6.035532	2.277075
14	538	10515	10304	17516	768	32319	0	4.840175	2.188689
15	510	10881	11599	16499	975	31458	0	4.623041	2.090619
16	513	10912	10887	18021	526	33907	0	4.637707	2.09725
17	455	13070	13667	22639	643	29066	0	3.997645	1.807581
18	478	12006	12288	18710	435	35276	0	4.211046	1.904431

ตารางผนวกที่ ก1 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
19	327	22189	14735	44568	844	77812	0	2.389163	1.080385
20	493	11336	10965	17815	1347	23213	0	4.458553	2.016034
21	457	12633	12316	20629	393	37920	0	4.029591	1.822438
22	485	11593	12771	18281	391	33220	0	4.352235	1.968592
23	525	10771	10899	18041	425	27247	0	4.716515	2.132986
24	466	12255	11587	20508	493	24812	0	4.141376	1.872881
25	500	11406	11444	17974	490	35573	0	4.457997	2.016251
26	459	12586	12801	20052	422	36296	0	4.030948	1.822896
27	436	12973	16175	19593	867	37009	0	3.90555	1.766361

ตารางผนวกที่ ก1 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
28	454	12814	11581	21715	485	39229	0	3.981408	1.800496
29	476	12145	11816	19892	632	38066	0	4.178297	1.889519
30	437	13407	13921	22111	495	39896	0	3.769028	1.704565

ดังนั้น ฟังก์ชันการทำงานแบบย่อแบบที่ 1 สามารถรองรับจำนวนผู้ใช้ได้เฉลี่ย 469.2 ≈ 469 คน, มีค่าเบี่ยงเบนมาตรฐาน เท่ากับ 74.83932

ตารางผนวกที่ ก2 ผลการทดลอง 30 ครั้งของฟังก์ชันการทำงานแบบย่อแบบที่ 2

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
1	580	9556	9004	16366	833	18037	0	6.296078	2.301518
2	405	13659	14839	22958	1079	36355	0	3.724309	1.618474
3	575	9905	8867	18344	829	20220	0	6.164686	2.244419
4	574	9656	9423	15062	1434	16238	0	6.281804	2.295315
5	357	16137	18098	20758	2239	24577	0	3.115319	1.353825
6	600	9170	9907	15191	825	16563	0	6.52588	2.401383
7	628	8669	9188	13991	655	15459	0	6.802029	2.521389
8	610	8897	8261	14168	876	15431	0	6.657	2.458364
9	607	9105	8621	15621	1080	17187	0	6.576328	2.423306

ตารางผนวกที่ ก2 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
10	552	9817	11277	14413	1167	16002	0	6.157914	2.241476
11	496	11137	11828	17576	973	22861	0	4.550459	1.977494
12	411	15499	14725	26248	1378	30832	0	3.315452	1.440797
13	580	9388	9666	16267	414	18447	0	6.358067	2.328457
14	609	8784	8574	14129	1026	15667	0	6.749188	2.498426
15	631	8594	8177	14546	489	15644	0	6.90437	2.565864
16	627	8771	9087	14721	332	15627	0	6.759162	2.502761
17	605	8945	9254	14081	399	15186	0	6.611672	2.438666
18	614	9307	8812	17032	594	19954	0	6.536519	2.406007

ตารางผนวกที่ ก2 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
19	524	10581	10662	17170	681	22947	0	4.791558	2.082269
20	600	9174	9303	15143	943	16342	0	6.486968	2.384474
21	642	8518	8448	14921	353	17346	0	6.960173	2.590114
22	593	9234	10469	15278	299	16572	0	6.460707	2.373061
23	593	9073	10690	13658	901	15020	0	6.572837	2.421789
24	640	8483	8150	14603	384	16492	0	6.960308	2.590173
25	624	8860	9337	15201	265	16770	0	6.678457	2.467689
26	602	9030	9176	14462	289	15336	0	6.550996	2.412298
27	563	10209	9294	18539	353	21316	0	6.077378	2.206478

ตารางผนวกที่ ก2 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
28	551	10007	10113	16385	732	19316	0	6.077312	2.206449
29	594	9313	9968	14721	373	16256	0	6.448392	2.367709
30	593	9347	8835	16362	443	18763	0	6.455131	2.370638

ดังนั้น ฟังก์ชันการทำงานแบบย่อแบบที่ $_2$ สามารถรองรับจำนวนผู้ใช้ได้เฉลี่ย 572.7$\approx$ 572 คน, มีค่าเบี่ยงเบนมาตรฐาน เท่ากับ 70.02479

ตารางผนวกที่ ก3 ผลการทดลอง 30 ครั้งของฟังก์ชันการทำงานแบบย่อแบบที่ 3

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
1	1838	2691	2397	5025	235	10588	0	18.74994	7.965061
2	1882	2666	2465	4891	119	9517	0	18.72544	7.954653
3	1631	3192	3026	6196	275	13874	0	16.67501	6.658816
4	1616	3088	2950	5539	113	12529	0	16.4236	6.976822
5	1636	3041	2549	5960	89	11723	0	16.71571	7.100912
6	1778	2795	2682	5031	142	8174	0	18.04527	7.665713
7	1761	2863	2633	5122	115	11270	0	17.56977	7.463719
8	1806	2766	2252	5675	108	14303	0	18.24961	7.752521
9	2047	2447	2235	4384	149	8201	0	20.46673	8.694361

ตารางผนวกที่ ก3 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
10	1707	2977	2840	5365	147	9435	0	17.08897	7.259474
11	1519	3329	3014	6442	134	17725	0	16.36143	6.525609
12	1807	2752	2637	4992	135	8775	0	18.30188	7.774726
13	1441	3502	3352	5588	166	10711	0	14.48256	6.152258
14	1493	3360	3215	6052	256	10939	0	16.06422	6.399352
15	1921	2621	2375	4549	205	8862	0	19.27941	8.189982
16	1951	2643	2699	4184	196	9176	0	19.55831	8.308461
17	1578	3263	3109	6166	89	11840	0	16.66657	6.655234
18	1649	3108	2805	6000	148	11238	0	16.46892	6.996074

ตารางผนวกที่ ก3 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
19	1757	2903	2721	5378	152	11617	0	17.55911	7.459194
20	1942	2598	2433	4471	219	8159	0	19.61775	8.333711
21	1712	2963	3157	5260	108	8497	0	17.1135	7.269894
22	1774	2847	2536	5504	128	10086	0	17.8408	7.578856
23	1868	2717	2671	4999	96	10281	0	18.57967	7.892731
24	1685	3020	2635	5921	75	12429	0	16.78906	7.13207
25	1975	2532	2633	4254	127	8452	0	19.94708	8.473612
26	1761	2886	2661	5322	61	11248	0	17.60419	7.478343
27	1854	2687	2427	4875	166	9775	0	18.74659	7.963638

ตารางผนวกที่ ก3 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
28	1739	4665	3021	5359	84	65416	0	11.09162	4.711773
29	1841	2666	2495	4778	149	8201	0	18.88031	8.020444
30	1682	2977	2639	5759	126	10687	0	17.02963	7.234269

ดังนั้น ฟังก์ชันการทำงานแบบย่อแบบที่ 3 สามารถรองรับจำนวนผู้ใช้ได้เฉลี่ย 1756.033≈ 1755 คน, มีค่าเบี่ยงเบนมาตรฐาน เท่ากับ 146.5532

ภาคผนวก ข

ผลการทดลองเมื่อกำหนดเงื่อนไขที่ยอมรับได้สำหรับระบบอย่างไม่เหมาะสม

ผลการทดลองเมื่อกำหนดเงื่อนไขที่ยอมรับได้สำหรับระบบอย่างไม่เหมาะสม

ก่อนที่จะได้ผลการทดลองของฟังก์ชันการทำงานแบบย่อทั้ง 3 รูปแบบ เราได้ทำการทดสอบฟังก์ชันการทำงานแบบย่อแบบที่2 และฟังก์ชันการทำงานแบบย่อแบบที่ 3 ซึ่งเราได้กำหนดค่าที่ยอมรับได้ไม่เหมาะสม จึงส่งผลให้ค่า SD ของฟังก์ชันการทำงานแบบย่อแบบที่ 3 ค่อนข้างสูง เราได้ทำการทดสอบประสิทธิภาพ ตามกระบวนการ 7 ขั้นตอนของการทดสอบประสิทธิภาพ พร้อมทั้งวิเคราะห์ผลการทดลองได้ ดังนี้

กำหนดสภาวะแวดล้อมในการทดสอบ

สภาวะแวดล้อมในการทดสอบประกอบไปด้วยคอมพิวเตอร์ 2 เครื่อง เชื่อมต่อกันผ่านทางสาย LAN เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเครื่องแม่ข่ายใช้ มีคุณสมบัติ คือซีพียู Core2Duo ความเร็ว 2.16 GHz และหน่วยความจำหลักขนาด 2 GB, เครื่องที่สองทำหน้าที่เป็นเครื่องผู้ใช้ มีคุณสมบัติ คือ ซีพียู AMD E-450 ความเร็ว 1.65 GHz และหน่วยความจำหลักขนาด 2 GB ซึ่งจะใช้โปรแกรม JMeter ยิงกลุ่มผู้ใช้ไปยังเครื่องแม่ข่าย และดูผลลัพธ์

กำหนดเงื่อนไขที่ยอมรับได้

เรากำหนดผลลัพธ์จะขัดกับเงื่อนไขที่ยอมรับได้ เมื่อ (1) ภายใน 1 วินาที เวลาในการตอบสนองต่อผู้ใช้มากกว่า 1000 มิลลิวินาที และ Throughput น้อยกว่า 5 หรือ (2) ภายใน 10 วินาที ใช้หน่วยประมวลผลเกินกว่า 50% และอัตราความผิดพลาดมากกว่า 50%

วางแผนและออกแบบการทดลอง

เราจำลองกลุ่มของผู้ใช้จากเครื่องผู้ใช้ส่งไปยังเครื่องแม่ข่าย โดยเพิ่มจำนวนผู้ใช้ไปเรื่อยๆ จนกระทั่งการประมวลผลของระบบขัดกับข้อกำหนดเงื่อนไขที่ยอมรับได้ในข้อ 2 เราทำการทดสอบทั้ง 3 ระบบ แต่ละระบบเราทำการทดสอบ 30 ครั้ง แล้วนำผลหาค่าเฉลี่ย เพื่อเปรียบเทียบประสิทธิภาพว่าระบบที่ใช้ฟังก์ชันการทำงานแบบย่อแบบใดสามารถรองรับจำนวนผู้ใช้ได้มากที่สุด

ติดตั้งสภาวะแวดล้อมในการทดสอบ

ในเครื่องแม่ข่าย เราติดตั้ง Apache TOMCAT 7.0.11 เป็นเว็บเซิร์ฟเวอร์เพื่อรองรับจำนวนผู้ใช้, ใช้ MySQL เป็นฐานข้อมูล ระบบปฏิบัติการที่ใช้บนเครื่องแม่ข่าย คือ Windows Vista Ultimate ในเครื่องผู้ใช้ เราจำลองกลุ่มผู้ใช้มาจากโปรแกรม JMeter add-ins และระบบปฏิบัติการที่ใช้คือ Windows 7

กำหนดค่าต่างๆ ตามการทดลองที่ได้ออกแบบไว้

อย่างแรก เรากำหนด Thread Group คือ จำนวนผู้ใช้กลุ่มแรกมี 100 คน และค่อยๆเพิ่มขึ้นทุกๆ 10 คน ในทุกๆ 5 วินาที เงื่อนไขที่ยอมรับได้ ที่ถูกกำหนดไว้ใน ข้อ 2 หลังจากนั้น เราก็กำหนด IP Address, port และตัวแปรสุ่มต่างๆ เพื่อส่งไปยังเครื่องแม่ข่าย และสุดท้ายเพิ่ม Aggregate Report เพื่อดูผลการทดลองว่าสามารถรองรับจำนวนผู้ใช้ได้มากสุดเท่าใด

ทำการทดลอง

ฟังก์ชันการทำงานแบบย่อทั้ง 2 รูปแบบจะถูกเข้าไปทดสอบ 30 ครั้ง ภายใต้สภาวะแวดล้อมที่ได้กำหนดไว้ และนำผลของแต่ละครั้งมาหาค่าเฉลี่ยเพื่อเปรียบเทียบว่าฟังก์ชันการทำงานแบบย่อแบบใดสามารถรองรับจำนวนผู้ใช้ได้มากที่สุด โดยไม่ขัดกับเงื่อนไขและการยอมรับที่กำหนดไว้ในข้อ 2

ตารางผนวกที่ ข1 ผลการทดลองทั้ง 30 ครั้งของฟังก์ชันการทำงานแบบย่อแบบที่ 2 และ 3 (กำหนดค่าที่ยอมรับได้ไม่เหมาะสม)

ฟังก์ชันการทำงานแบบย่อ	ข้อมูลที่ถูกเขียนอย่างเดียว	ข้อมูลที่ถูกเขียนครั้งเดียว	ข้อมูลที่ถูกเขียนหลายครั้ง	ค่าสูงสุดของจำนวนผู้ใช้ที่สามารถรองรับได้สูงสุด
แบบที่ 2	688	312	472.70	94.82
แบบที่ 3	1846	436	844.37	360.34

วิเคราะห์ผลการทดลอง

ตารางผนวกที่ข1 แสดงให้เห็นผลลัพธ์ของการทดสอบประสิทธิภาพ คือ ฟังก์ชันการทำงานแบบย่อแบบที่ 2สามารถรองรับจำนวนผู้ใช้เฉลี่ยได้ 844 คน ส่วนฟังก์ชันการทำงานแบบย่อแบบที่ 3 สามารถรองรับจำนวนผู้ใช้เฉลี่ยได้ 472 คน ถ้าเราทำการปรับจากฟังก์ชันการทำงานแบบย่อแบบที่ 2 ให้เป็นฟังก์ชันการทำงานแบบย่อแบบที่ 3 แล้ว ประสิทธิภาพจะเพิ่มขึ้น 179% แต่เนื่องจากค่า SD ของฟังก์ชันการทำงานแบบย่อแบบที่ 3 ค่อนข้างสูงมาก เราสันนิษฐานว่าน่าจะมาจากการที่กำหนดค่าที่ยอมรับได้ที่ไม่เหมาะสม

ตารางผนวกที่ ข2 ผลการทดลอง30 ครั้งของฟังก์ชันการทำงานแบบย่อแบบที่ 2

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
1	620	7241	5806	14321	787	16754	0	6.978995	3.032864
2	545	8413	6496	17635	982	25614	0	6.113224	2.656626
3	482	9854	7756	19210	2991	34548	0	6.206308	2.262507
4	312	17956	7666	44075	1845	69015	0	3.059183	1.32943
5	395	13629	9352	28770	3097	47379	0	3.773369	1.639794
6	401	12548	7571	30480	2221	45852	0	4.17617	1.814839
7	458	10580	7239	23573	1363	38976	0	4.880752	2.12103
8	411	11777	8276	26373	2753	40092	0	4.394594	1.90976
9	477	13167	7937	35241	1331	48978	0	4.390123	1.907817

ตารางผนวกที่ ข2 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
10	387	18057	9187	47418	1483	66828	0	3.270708	1.421353
11	625	7110	5899	12738	1593	18422	0	7.188866	3.124068
12	560	7325	5115	16017	1311	26729	0	6.821532	2.964435
13	568	8187	6345	17773	1369	32138	0	6.124385	2.661476
14	410	13924	7188	33041	1814	52676	0	3.824912	1.662193
15	339	16314	9393	41406	1993	59785	0	3.261999	1.417568
16	334	21347	12949	52968	2282	76235	0	2.846769	1.237121
17	529	9469	6240	20638	648	36243	0	6.518752	2.398286
18	573	10840	5992	29388	759	43252	0	6.535644	2.405626

ตารางผนวกที่ ข2 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
19	464	12017	6331	28976	1317	44007	0	4.611134	2.003862
20	435	9506	6048	22698	1140	38097	0	6.284064	2.296297
21	394	19105	10380	49432	1496	72877	0	3.33689	1.450114
22	420	11350	7020	24785	1157	40271	0	4.505568	1.957986
23	422	13846	6763	36898	1067	55226	0	3.988017	1.733074
24	497	11410	7018	26463	1427	46763	0	4.904863	2.131508
25	460	12065	8099	28164	1107	41817	0	4.718965	2.050722
26	397	14735	8282	37099	1102	50011	0	3.826027	1.662678
27	688	6454	6109	10951	1184	13466	0	7.879336	3.424126

ตารางผนวกที่ ข2 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
28	620	7592	5681	15720	1217	25884	0	6.929542	3.011373
29	521	9713	5905	22273	1342	40902	0	6.510195	2.394567
30	437	14048	8047	36462	1517	53050	0	3.969696	1.725112

ดังนั้น ฟังก์ชันการทำงานแบบย่อแบบที่ 2 (แบบกำหนดค่าที่ยอมรับได้ไม่เหมาะสม) สามารถรองรับจำนวนผู้ใช้ได้เฉลี่ย 472.7≈ 472 คน, มีค่าเบี่ยงเบนมาตรฐาน เท่ากับ 94.8233

ตารางผนวกที่ ข3 ผลการทดลอง 30 ครั้งของฟังก์ชันการทำงานแบบย่อแบบที่ 3

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
1	1664	2430	2115	3855	728	8842	0	20.57496	8.74034
2	1242	3316	3013	6463	1014	14014	0	16.07501	6.403935
3	1038	4153	3170	8383	412	14855	0	12.42444	6.277961
4	607	7620	4671	18064	502	33540	0	6.785383	2.882462
5	436	11197	5616	27869	1386	51897	0	4.505855	1.914108
6	584	9843	4333	27854	686	47775	0	6.716187	2.428263
7	690	7640	3632	20198	1016	38988	0	7.205815	3.061064
8	795	5738	3386	14980	1014	30213	0	9.235273	3.923187
9	742	6672	4227	16031	1049	26639	0	7.932605	3.369808

ตารางผนวกที่ ข3 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
10	706	6745	4256	16346	1039	30153	0	7.756707	3.295086
11	554	8292	3512	25729	1048	46498	0	6.990744	2.544896
12	1846	2181	1870	3459	1030	6787	0	23.29837	9.897258
13	1214	3346	3026	6475	983	12762	0	16.25356	6.479782
14	1102	4103	3106	8244	1036	16865	0	12.63356	6.366795
15	635	7101	3603	18149	569	35415	0	7.228641	3.070761
16	587	9239	3516	27239	1029	48373	0	6.690797	2.417477
17	539	9342	4249	26105	1020	44391	0	6.65523	2.402368
18	744	7141	3489	23721	1020	43437	0	7.715441	3.277556

ตารางผนวกที่ ข3 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
19	699	6797	3582	17637	1002	35715	0	7.529488	3.198562
20	703	8229	3678	27442	1026	46876	0	6.660287	2.829321
21	683	6864	3671	18079	1013	32855	0	7.706977	3.27396
22	556	10103	4559	30166	1093	52572	0	6.437919	2.310053
23	606	9342	5458	26114	1014	40156	0	6.976626	2.538899
24	701	8222	3907	25676	1024	47254	0	6.694553	2.843877
25	610	7301	3670	18582	1008	39460	0	7.002158	2.97455
26	733	7644	4148	23843	1014	38742	0	7.219186	3.066744
27	1523	2711	2156	4826	1013	12894	0	18.89484	8.026618

ตารางผนวกที่ ข3 (ต่อ)

ครั้งที่	Samples	Average	Median	90% line	Min	Max	Error	Throughput	KB/sec
28	1325	3217	2648	6146	1012	14475	0	16.98292	6.789619
29	868	5536	3069	13225	1006	26906	0	9.774885	4.152417
30	599	11278	3227	46428	1030	74492	0	4.948041	2.101951

ดังนั้น ฟังก์ชันการทำงานแบบย่อแบบที่ 3 (แบบกำหนดค่าที่ยอมรับได้ไม่เหมาะสม) สามารถรองรับจำนวนผู้ใช้ได้เฉลี่ย 844.3667 ≈ 844 คน, มีค่าเบี่ยงเบนมาตรฐาน เท่ากับ 360.3399

ภาคผนวก ค

บทความวิทยานิพนธ์ที่ได้รับการตีพิมพ์เผยแพร่

Light-weight Operation of a Failover System for Cloud Computing

Pakpoom Mookdarsanit
Department of Computer Science
Faculty of Science, Kasetsart University
Bangkok 10900, THAILAND
g5314450089@ku.ac.th

Sethavidh Gertphol
Department of Computer Science
Faculty of Science, Kasetsart University
Bangkok 10900, THAILAND
fscisvg@ku.ac.th

***Abstract*—Availability is one of the biggest challenges which slow down the adoption of Cloud computing in the IT industry. A failover system can be designed to use as a backup in case of Cloud failure or unavailability. In this paper, we introduce the method for designing a local failover system for Cloud. Since a failover system has less resource and limited scalability; it cannot handle all workloads previously on the Cloud. We overcome this drawback by adapting the full operation performed in the Cloud into a light-weight operation optimized for a failover system. A light-weight operation consumes less resource but also has reduced quality of service (QoS) compared to the full operation. We also define a write type of data according to change and future use of that data. We adapt the full operation into a light-weight one by reducing the number of write types or changing one type to another. In other words, we reduce the quality of service of the system in order to serve more requests. Experimental result shows that a system with full operation can sustain an average of 472 maximum numbers of requests, and a system with light-weight operation can sustain 844 requests, an improvement of 179%.**

Keywords-Cloud Computing; Availability; Fault Tolerance; failover system; Adaptive Quality of Service; Load Testing

I. INTRODUCTION

Cloud computing is a new type of infrastructure for computing environment. Cloud users do not have to know about the physical infrastructure. All large physical resources are shared among users, but logically allocating as a virtual infrastructure for individual user. Transition to virtual infrastructure reduces the cost for hardware and software management, which is the main benefit for the Cloud user. Cloud computing provides hardware and/or software resources as services, which can be easily scaled to optimize for the user needs. Many services in Cloud computing are charged according to pay-per-use model [1, 2].

However, there are many challenges, such as security, availability, SLA guarantee, that must be overcome before Cloud Computing is widely adopted by mainstream IT industry. Availability is one issue that IT professionals are concern the most, especially in countries where basic IT infrastructure such as networking is not stable or mature enough. The Cloud running by experienced IT staffs may be working fine, but the access networks to the Cloud running by local, less reputable IT staffs may cause availability issue. When the Cloud becomes unavailable, all operations and data running on the Cloud are considered failed as well. Cloud users should find a solution to continue providing services in the face of this failure.

One technique that can be used to continue services in case of failure is a failover system which operates as a backup for the main system and comes online when the main system fails. Data from the main system must also be back-upped for the failover system to use. However, designing a failover system for the Cloud brings other challenges because the local resource a company has is minuscule comparing to the Cloud. The failover system cannot sustain the same amount of workload executed on the Cloud.

In this paper, we propose a novel technique to design a failover system for the Cloud, providing another resource to automatically switch to, if the Cloud becomes unavailable. The goal of a

failover system policy is high throughput for user requests. Since a failover system has less resource and limited scalability; it cannot handle all workloads previously on the Cloud. We overcome this drawback by adapting the full operation performed in the Cloud into a light-weight operation optimized for a failover system. A light-weight operation consumes less resource but also has reduced quality of service compared to the full operation. We also define a write type of data according to change and future use of that data. We adapt the full operation into a light-weight one by reducing the number of write types or changing one type to another. In other words, we reduce the quality of service of the system in order to serve more requests. The experiment compares the maximum number of requests a full operation and the light-weight operation can sustain. Both operations are evaluated by performing a load testing using Apache JMeter [12]. Each operation was tested 30 times. The result is that a system with full operation can sustain an average of 472 maximum numbers of requests, and a system with light-weight operation can sustain 844 requests, an improvement of 179%.

The organization of this paper is as follows. Some related works are briefly described in Section II. Section III describes an idea for adapting full operation into light-weight operation of failover system, and classifying write types into three types. Section IV shows the result of load testing.

II. RELATED WORKS

Availability is one of the biggest challenges which slow down the adoption of Cloud computing in the IT industry [3]. There are several techniques to improve availability and fault-tolerance in the literature. For example, authors in [4] intentionally injected virtual fault types into a system and checking for nodes that fail. Replicating information from the failed nodes into their neighbor beforehand helps increase fault-tolerance. In [5], replication management of the system is distributed into many groups of server replicas. Each group has a tool for governing its communication in the group and with other group. Research in [6] classified the availability of data in mobile systems into three types according to different replication method. Online availability is when a user can access data from any device connected to the server. Offline availability is when a user can access data from the set of connected (nearby) devices but not from the server. Finally, ownership availability is when a user can access data from individual device only. Availability type in [6] can be compared with quality of service in our research where each type has different QoS depending on availability of data.

Adapting QoS to achieve higher performance or to handle resource restriction is a classic technique in many fields. In [7], a solution for adapting web content to workload conditions to alleviate overload is proposed. Some web content adaption techniques are introduced such as, substituting low quality image, reducing the number of embedded objects per page, and reducing local link. The system in [8] is based on quality adaption which automatically switches from original version to reduced QoS version when the server is overload. The Scalable VDO Coding (SVC) standard [9] enables quality adaptation for VDO including graceful degradation in loss transmission environments as well as bit rate, format, and power adaptation.

III. FAILOVER SYSTEM CONCEPT

We describe our system model where local resources in a failover system act as a backup for the Cloud. The system always uses Cloud resources during normal operation, however, when the Cloud goes off-line, the system switches to the failover system. Compared to the Cloud, a failover system usually has less resources and limited scalability, so it cannot handle all of the workload previously executing on the Cloud.

A. Workload of the system

The amount of system workload depends on two factors: the number of client requests and the operation performed by each request. We define “full operation” as the work performed as part of the functional requirement of the system. For example, an operation for booking a movie ticket

consists of: (1) checking whether there are any vacant seats, (2) checking the seat number is available or not, (3) booking a movie ticket with a seat number, and finally (4) decreasing the number of vacant seats by one.

A more "light-weight operation" can be defined by reducing some functional requirements of the system. Continuing from the example, a light-weight operation of booking a movie ticket does not check how many vacant seats are available, but accepts the user booking right away. It also does not have to decrease the number of vacant seats. At some later time, the system tabulates all requests from users and issues movie tickets in a first-come-first-serve basis. Note that after the completion of this light-weight version of the operation, the system cannot inform a user whether he will have a movie ticket or not. It can guarantee only that it has received user's request and will issue movie tickets later. So, we can say that this light-weight operation has a lower quality of service (QoS) than the full operation.

Figure 1 shows the Cloud has much more capacity and its amount of workload can be much greater than the capacity of a failover system. When the Cloud fails, the workload must be reduced in order for the failover system to work. Since workload of a system depends on the number of user requests and type of operation, we can either reduce the number of user requests or change from full operations to light-weight operations. Because the number of requests is an external factor not under our direct control, it is easier to change the type of operations performed in our own system. So, operation performed by each request in failover system is less than in the Cloud but failover system can serve number of requests as the Cloud. In other words, we reduce the quality of service of the system in order to serve more requests.

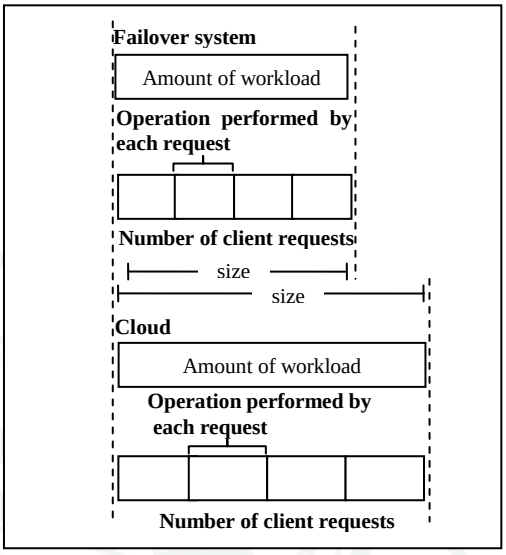


Figure 1. Amount of workload in the Cloud and failover system

B. Effects of Reduced QoS

We adapt a full operation into a light-weight operation to reduce workload of the failover system and increase the number of requests. However, performing a light-weight operation results in a reduced Quality of Services for the users too. A service performed for the user becomes a partial service. For example, when a full operation for booking movie tickets completes resulting in a full QoS, the user can be sure that he owns a ticket. When a lighter weight operation completes with a reduced QoS, the user may not have a ticket yet but can be sure that the tickets will be issued later by a first-come-first-serve manner. An even lighter weight operation may only guarantee that tickets will be issued later randomly.

C. Write types of light-weight operation

We define "write type" by the change performed on data and the future usage of that data after a light-weight operation has been executed. The future use of that data is defined within the same operation, i.e. how the data will be used by the same operation that will be initiated by other users. Also, one light-weight operation may use several data with differing write type. In this paper, the effect of a light-weight operation and its QoS together with its

write types on system workload is investigated. Each write type requires different methods for synchronization and/or backup between the failover system and the Cloud, however, we will not discuss synchronization in this paper.

We classify write type into three types: Multiple-write data, Write-once data and Write-only data. We also define "cell" as a place where data is stored. A cell may be a location in memory or a field in a database table. Within the same database table, each cell has different address.

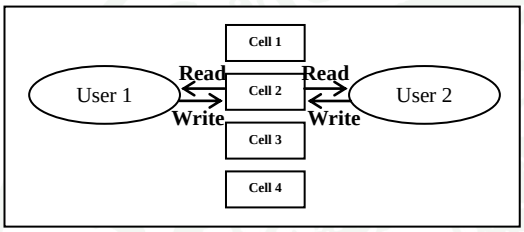


Figure 2. Multiple-write data write type

1) Multiple-write data: Data changing characteristics of this write type are: (1) reading previous data from the cell before writing a new result into the same cell, (2) it may be necessary to read data from other cells to calculate the result, and (3) when data is written, the data will be read or written again in the future.

For example, the number of vacant seats for booking movie ticket is a multiple-write data. A booking operation will check whether there are any vacant seats. If so the operation will book a ticket before updating the vacant seat value. After the data was updated, the vacant seats will be checked or updated again by the booking operation performed by other user in the future.

2) Write-once data: Data changing characteristics are: (1) before writing result into the cell, it necessary to read data from other cell for calculation, and (2) when data is written, it will be read only in the future.

For example, the seat number for booking a movie ticket is a write-once data. To book a seat, the system will check whether the seat number we need is available or not by reading a list of already booked seats. If the seat we want is not on the list, the system will book it by writing its number into the already-booked list. After the seat has been booked, the seat number will be checked (read) but never updated by other user in the future.

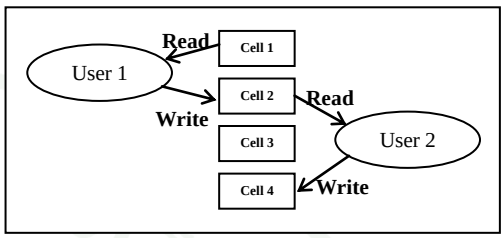


Figure 3. Write-once data

3) Write-only data: Data changing characteristics are: (1) before writing into the cell, it may or not be necessary to read data from other cell, and (2) when data is written, there is no future use by the same operation initiated by other users.

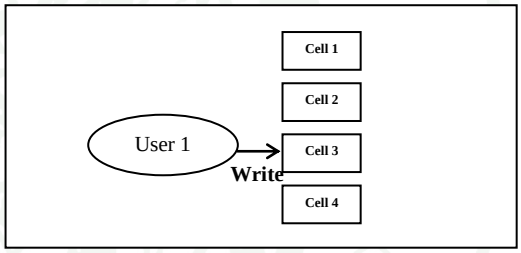


Figure 4. Write-only data

For example, the personal booking data for a movie ticket is a write-only data. When the system records that this person has booked a movie ticket, it will not be used by a booking operation in the future. It may be used by other operations (e.g. printing ticket operation) but the write type is defined on the same operation.

IV. Implementation and Evaluation

For a failover system to support more user requests, it must perform a light-weight operation with reduced QoS instead of a full operation executed on the Cloud. We can design a light-weight operation from the full operation by changing its write type. To evaluate the effect of a light-weight operation on system workload, we implemented and load tested two versions of booking movie ticket operations. One version is the full operation that can guarantee ownership of ticket after booking. Another version is a

light-weight operation that records booking requests from users and can guarantee only that the ticket will be issued in a first-come-first-serve basis.

Both systems were implemented as a web application using JSP and MySQL as a database. We evaluated both versions by performing load testing using JMeter with automatic ramp-up and stop Add-ins [10].

A. Light-weight operation for failover system

The full operation for booking a movie ticket consists of: (1) checking whether there are any vacant seats, (2) booking a movie ticket, and (3) decreasing the number of vacant seats by one. This operation has two write types: (1) the number of vacant seats is a Multiple-write data because after data is written, the cell will be read or written in the future, and (2) the booking data is a Write-only data because after data is written, there is no future use by other users.

The light-weight operation accepts the request for booking a movie ticket and records that request right away without checking the availability of seats. It thus does not have to update vacant seats value too. Note that this light-weight version of the operation cannot guarantee a user that he will have a movie ticket. This is a reduced quality of service compared with the full operation that can guarantee ownership of ticket after the full operation completes. The light-weight operation has only one write type: the booking data is a Write-only data because after data is written, there is no future use.

B. Load Testing

We evaluate the two testing sets according to 7 activities of core performance testing as described in [11].

1) Identify the test environment: The testing environment has two computers, connecting via a LAN. The server has 2.16 GHz Core2Duo CPU and 2GB of RAM. Another computer has a 1.65 GHz AMD E-450 CPU and 2GB of RAM, which run JMeter that generated virtual users to the server and getting the result.

2) Identify performance acceptance criteria: The acceptance criteria were violated when (1) within one second, a response time of any request was more than 1000 ms and throughput was less than 5 requests, or (2) within 10 second, CPU utilization is more than 50% and the Error rate is more than 50%.

3) Plan and design tests: We generated a number of requests from client to server, which automatically ramped-up until the acceptance criteria are not satisfied. We separately tested both versions 30 times and comparing the average of maximum number of requests that each version can sustain.

4) Configure the test environment: In the server computer, we configured Apache TOMCAT 7.0.11 as a web server for handling requests and using MySQL as a database. The operating system on the server was Windows Vista Ultimate. The client machine ran JMeter add-ins [10] to send requests to the server and receive results and running on Windows 7.

5) Implement the test design: First, we configured the Thread Group as follows: the first group of requests is 100 requests, with automatically ramp-up by 10 requests every 5 seconds. The acceptance criteria were configured as in section 4.2.2. Then, a server was added into the Thread Group with the IP address, port number and randomizing arguments to pass to server. Finally, the Aggregate Report tool was added into the Thread Group to gets the maximum number of requests.

6) Execute the test: Each testing set in the configuration environment was executed 30 times. Each time, the maximum number of requests that the server can processed without violating the acceptance criteria was recorded.

7) Analyze results, report and retest: Table I shows the result of load testing. The light-weight operation has only 1 data write type which is one write-only (booking data). The full operation has 2 data write types which is one multiple-write (number of vacant seats) and one write-only (booking data). It is therefore obvious that the full operation system does more work and has higher workload per request. The interesting question is how much more requests can be accepted if we instead use a light-weight operation system with reduced QoS.

TABLE I. MAXIMUM NUMBER OF REQUESTS WITHOUT VIOLATING ACCEPTANCE CRITERIA BASED ON 30 TEST ITERATIONS.

Operation	Max	Min	Average	SD
Light-weight	1846	436	844.37	360.34
Full	688	312	472.70	94.82

From Table I, the maximum number of requests averaged over 30 iterations for the light-weight operation is 844.37 requests while for the full operation is only 472.70 requests, an improvement of 179%. However, the standard deviation for the light-weight system is very high at 360.34 while the SD for the full operation is only 94.82. This high SD corresponds with the maximum value (among 30 iterations) of maximum requests for light-weight system at 1846 and minimum (among 30 iterations) of 436. We speculate that the high SD is due to parameters in the step of "Identify performance acceptance criteria" are not set appropriately, i.e. the acceptance criteria were violated when error rate is more than 50% within 10 second that might not be suitable identified. We will investigate this issue further.

V. CONCLUSION

This paper proposes a technique to design a local failover system for Cloud by adapting the full operation performed in the Cloud into a light-weight operation optimized for a failover system. This failover system uses fewer resources to perform light-weight operations with reduced quality of service instead of the full operations executed on the Cloud. We also defined 3 write-types for data – Multiple-Write, Write-Once, and Write-Only – depending on the change and future use of that data. We believe that a light-weight operation can be designed from the full operation by reducing in the full operation the number of write-type data and/or changing one write-type to another.

An experiment shows that a light-weight operation with one Write-Only when compared with a full operation having one Multiple-Write and one Write-Only can increase the maximum number of requests a system can sustain by 179%. This improvement can be achieved by trading off the quality of service; the light-weight operation cannot guarantee as much as the full operation. The reduced QoS should be further investigated in the context of application design. We will also continue research about the effect of write-types on workload in the future.

REFERENCES

[1] L. M. Vaquero, L Rodero-Merino, J. Caceres, and M. Linder, "A Break in the Clouds: Towards a Cloud Definition," ACM SIGCOMM Computer Communication Review, Vol.39 Issue1, pp. 50 – 55, January 2009.

[2] S. Ramgovind, M. M. Eloff and E. Smith, "The management of security in Cloud computing," Information Security for South Asia (ISSA), pp. 1-7, 2010.

[3] F. Gens, "New IDC IT Cloud Services Survey: Top Benefits and Challenges" IDC eXchange, 2009, [Online]. Available: http://blogs.idc.com/ie/?p=730 [Accessed: 6 September 2012].

[4] A. F. Zorzo and F. R. Meneguzzi, "An agent model for fault-tolerant systems," Proceedings of the 2005 ACM symposium on Applied Computing, pp. 60–65, 2005.

[5] S. Bora. "A Fault Tolerant System Using Collaborative Agents," F.A. Savacı (Ed.): TAINN 2005, LNAI 3949, pp. 211 – 218, 2006.

[6] C. Peery, T. D. Nguyen and F. M. Cuenca-Acuna, "Reducing the Availability Management Overheads of Federated Content Sharing Systems," 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06), pp. 5-17, 2006.

[7] T. F. Abdelzaher and N. Bhatti, "Web content adaptation to improve server overload behavior," The International Journal of Computer and Telecommunications Networking, 31(11-16), pp. 1563–1577, 1999.

[8] M. Gopshtein and D. G. Feitelson, "Trading off Quality for Throughput Using Content Adaptation in Web Servers," Proceedings of the 4th Annual International Conference on Systems and Storage, Proceedings of SYSTOR 2011: The Israeli Experimental Systems Conference, 2011.

[9] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 17, No. 9, pp. 1103-1119, 2007.

[10] N. Phasuk, JMeter Plugin Development for Automated Load Testing, Minor Thesis, Department of Computer Science, Faculty of Science, Kasetsart University, 2012.

[11] J.D. Meier, C. Farre, P. Bansode, S. Barber and D. Rea, "patterns & practices: Performance Testing Guidance for Web Applications" Microsoft Press: 2007, [Online]. Available: http://perftestingguide.codeplex.com [Accessed: 14 September 2012].

[12] "Apache: Apache JMeter", [Online]. Available: http://jmeter.apache.org/index.html [Accessed: 30 September 2012].

ประวัติการศึกษาและการทำงาน

ชื่อ –นามสกุล	นายภาคภูมิ มุกดาสนิท
วัน เดือน ปี ที่เกิด	30 กันยายน 2530
สถานที่เกิด	อำเภอธนบุรี จังหวัดกรุงเทพมหานคร
ประวัติการศึกษา	วศ.บ. (วิศวกรรมคอมพิวเตอร์) มหาวิทยาลัยมหิดล
ตำแหน่งหน้าที่การงานปัจจุบัน	อาจารย์ประจำ
สถานที่ทำงานปัจจุบัน	ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์