บทที่ 1

บทนำ

ความเป็นมาและความสำคัญของปัญหา

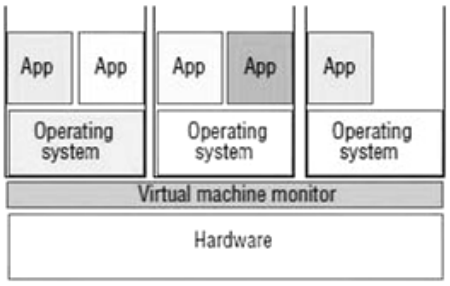
ระบบคอมพิวเตอร์ถูกพัฒนาไปอย่างรวดเร็วจนกลายเป็นระบบแบ่งเวลา (time-sharing system) เพื่อให้ผู้ใช้ (และแอพพลิเคชัน) จำนวนมากสามารถใช้งานระบบได้ในเวลาเดียวกัน อย่างไรก็ตาม การแบ่งเวลาคอมพิวเตอร์หนึ่งเครื่องทำให้เกิดปัญหา เช่น การใช้ระบบในทางที่ผิดอาจโดยตั้งใจหรือไม่ตั้งใจ สามารถทำให้คอมพิวเตอร์ทั้งเครื่องหยุดทำงานได้ การมีระบบคอมพิวเตอร์หลายระบบสามารถแก้ปัญหานี้ได้ ซึ่งมีประโยชน์ในแง่ของการแยกการทำงานออกจากกัน ในหลายสถานการณ์ควรจะมีงานบางอย่างที่รันบนระบบที่แยกจากกัน เช่น ข้อผิดพลาดในแอพพลิเคชันหนึ่งอาจจะสามารถรบกวนแอพพลิเคชันอื่นๆบนระบบเดียวกันได้ และประโยชน์ในด้านประสิทธิภาพ การให้แอพพลิเคชันรันบนระบบของมันเองจะทำให้มันสามารถใช้งานทรัพยากรของระบบได้เต็มที่ ซึ่งทำให้มีประสิทธิภาพดีกว่าการแบ่งระบบกับแอพพลิเคชันอื่น อย่างไรก็ตาม ค่าใช้จ่ายของระบบคอมพิวเตอร์หลายระบบอาจเป็นการสิ้นเปลือง และการมีเครื่องเพิ่มขึ้นทำให้ยากขึ้นที่จะทำให้เครื่องเหล่านั้นทำงานตลอดเวลา ในปี 1960 IBM เริ่มพัฒนาเวอร์ชวลแมชชีนเครื่องแรกที่ทำให้คอมพิวเตอร์หนึ่งเครื่องสามารถถูกแบ่งได้เหมือนกับมันเป็นคอมพิวเตอร์หลายเครื่อง IBM พัฒนาเวอร์ชวลแมชชีนในลักษณะที่เป็นเมนเฟรมคอมพิวเตอร์แบบแบ่งเวลาซึ่งมีค่าใช้จ่ายสูง โดยทั่วไปเครื่องเมนเฟรมหนึ่งเครื่องจะถูกใช้สำหรับการพัฒนาและการใช้งานแอพพลิเคชัน การพัฒนาแอพพลิเคชันบนระบบเดียวกับแอพพลิเคชันที่ต้องการใช้งานในขณะที่แอพพลิเคชันอื่นๆอยู่ในขั้นตอนการนำไปใช้งานจริงบนระบบนั้น ถือว่าเป็นการปฏิบัติที่ไม่ดี อาจทำให้ต้องรีบูทหลายครั้งหรือทำให้ระบบขาดเสถียรภาพ ซึ่งสิ่งเหล่านี้ไม่ควรจะไปแทรกแซงแอพพลิเคชันที่ถูกพัฒนาเสร็จไปแล้ว (Rose, 2004, pp. 1-2)

ตามแนวคิดเกี่ยวกับระบบเวอร์ชวลแมชชีนที่อิมพลีเมนต์บนระบบคอมพิวเตอร์ที่สามารถเวอร์ชวลไลซ์ได้ของ Popek และ Goldberg (1974, p. 413) เวอร์ชวลแมชชีน (Virtual Machine หรือ VM) คือเครื่องจำลองที่เหมือนกับเครื่องจริง ซึ่งเป็นระบบที่แยกออกมาต่างหากและมีประสิทธิภาพ จากภาพที่ 1.1 ส่วนหนึ่งของซอฟต์แวร์ที่เรียกว่าเวอร์ชวลแมชชีนมอนิเตอร์ (Virtual Machine Monitor หรือ VMM) มีลักษณะที่สำคัญ 3 ประการคือ 1) มีสภาพแวดล้อมสำหรับโปรแกรมที่เหมือนกับเครื่องจริง 2) กรณีที่แย่ที่สุดของโปรแกรมที่รันในสภาพแวดล้อมนี้คือ

ความเร็วที่ลดลงเพียงเล็กน้อย และ 3) สามารถควบคุมทรัพยากรระบบอย่างเต็มที่ โดยเวอร์ชวลแมชชีนมอนิเตอร์จะสร้างเวอร์ชวลโปรเซสเซอร์และส่วนอื่นๆของอุปกรณ์ระบบ เช่น อุปกรณ์ไอโอที่เก็บข้อมูล หน่วยความจำ ให้กับเวอร์ชวลแมชชีน ซึ่งคุณสมบัติที่สำคัญของสภาพแวดล้อมที่ทำการเวอร์ชวลไลซ์นี้คือการแยกออกจากกันระหว่างเวอร์ชวลแมชชีนที่มันจะต้องจัดการ ดังนั้นปัญหาที่เกิดขึ้นในเวอร์ชวลแมชชีนเครื่องหนึ่งจะไม่มีผลกระทบต่ออีกเครื่องหนึ่ง จากทฤษฎีของ Popek และ Goldberg สิ่งสำคัญที่สุดที่จะต้องมีในสถาปัตยกรรมคอมพิวเตอร์เพื่อที่จะทำให้มันสามารถเวอร์ชวลไลซ์ได้ คือการดักจับคำสั่งพิเศษ หมายความว่าเมื่อเกสเวอร์ชวลแมชชีน (ขณะที่รันอยู่บนโปรเซสเซอร์จริงโดยตรง) พยายามที่จะเอ็กซิคิวต์คำสั่งพิเศษ โปรเซสเซอร์จะหยุดและคืนการควบคุมไปที่เวอร์ชวลแมชชีนมอนิเตอร์ เพื่อที่มันจะได้ตัดสินใจว่าจะเอ็กซิคิวต์คำสั่งหรือไม่หรือจะจำลองคำสั่งด้วยวิธีการอื่นๆ (Rose, 2004, pp. 3-4)

ภาพที่ 1.1

เวอร์ชวลแมชชีนมอนิเตอร์ (Virtual Machine Monitor หรือ VMM)



ที่มา: “Virtual machine monitors: current technology and future trends,” โดย Rosenblum, M., & Garfinkel, T., 2005, *Computer, 38*(5), น. 40.

Kernel-based Virtual Machine (KVM) คือระบบที่มีการขยายเวอร์ชวลไลเซชันเพื่อเพิ่มความสามารถของเวอร์ชวลแมชชีนมอนิเตอร์ (หรือไฮเปอร์ไวเซอร์) ให้กับลินุกซ์ การใช้เควีเอ็มจะทำให้สามารถสร้างและรันเวอร์ชวลแมชชีนหลายๆเครื่องได้ เวอร์ชวลแมชชีนเหล่านี้จะเหมือนกับเป็นลินุกซ์โปรเซสปกติและจะรวมเข้ากับส่วนที่เหลือของระบบ (Kivity, Kamay, Laor, Lublin, & Liguori, 2007, p.225) โดยเควีเอ็มถูกพัฒนาขึ้นบนพื้นฐานของคีมู (QEMU) (Bellard, 2008) ซึ่งเป็นแมชชีนอีมูเลเตอร์ที่สามารถทำการจำลองระบบได้อย่างรวดเร็ว เครื่องมือที่สำคัญอย่างหนึ่งของเควีเอ็มไฮเปอร์ไวเซอร์คือ ไลฟ์ไมเกรชัน (live migration) ซึ่งหมายถึงความสามารถ

ในการย้ายเวอร์ชวลแมชชีนจากเครื่องหนึ่งไปอีกเครื่องหนึ่งโดยเกือบจะไม่ขัดจังหวะการทำงานของเวอร์ชวลแมชชีน นั่นคือเวอร์ชวลแมชชีนจะสามารถทำงานตามปกติของมันต่อไปได้ในขณะที่ทำการย้ายสถานะ เพื่อความเหมาะสมของจำนวนงานที่โหลดในแต่ละเครื่องและความต้องการด้านประสิทธิภาพ โปรโตคอลสำหรับไมเกรชันมี 2 แบบคือโปรโตคอลที่ใช้ไมเกรทเวอร์ชวลแมชชีนระหว่างเครื่องได้โดยการใช้ทีซีพีซ็อกเก็ต และโปรโตคอลที่ทำการบันทึกอิมเมจลงไฟล์ (ไมเกรทสถานะลงไฟล์หรือเซ็คพอยน์) ซึ่งจะอาศัยการทำงานของ savevm ที่คีมูมีให้ โดย savevm เป็นเครื่องมือที่ใช้สร้างสแนปช็อตของเวอร์ชวลแมชชีนซึ่งประกอบด้วยสถานะของซีพียู หน่วยความจำ สถานะของดีไวซ์ และส่วนของดิสก์ที่เขียนได้ทั้งหมด ความสามารถของการไมเกรทสถานะลงไฟล์ของเควีเอ็มแตกต่างจาก savevm ของคีมูคือ savevm จะเก็บข้อมูลสถานะของเวอร์ชวลแมชชีนในดิสก์อิมเมจที่มีรูปแบบเป็น qcow2 ดังนั้น การเรียกสแนปช็อตของเวอร์ชวลแมชชีนขึ้นมาจะต้องทำภายในคีมูมอนิเตอร์ แต่การไมเกรทสถานะลงไฟล์ของเควีเอ็มจะบันทึกสถานะของเวอร์ชวลแมชชีนลงในไฟล์ปกติ ทำให้สามารถเรียกเวอร์ชวลแมชชีนขึ้นมาได้ทางคอมมานด์ไลน์นอกคีมูมอนิเตอร์ อย่างไรก็ตาม การเซ็คพอยน์เวอร์ชวลแมชชีนโดยใช้ไมเกรชันโปรโตคอลนี้จะต้องทำการหยุดเวอร์ชวลแมชชีนโดยใช้คำสั่งในคีมูมอนิเตอร์ก่อนที่จะใช้คำสั่งไมเกรทเพื่อทำเซ็คพอยน์ ทำให้การทำงานไม่เป็นไลฟ์ไมเกรชันเหมือนกับการไมเกรทระหว่างเครื่อง

ผลการทดลองเซ็คพอยน์ด้วยวิธีการไลฟ์ไมเกรชันที่อิมพลีเมนต์ในงานวิจัยนี้พบว่าเทคนิคไลฟ์ไมเกรชันสามารถนำมาใช้กับการไมเกรทลงไฟล์หรือการเซ็คพอยน์เพื่อปรับปรุงประสิทธิภาพการทำงานของแอพพลิเคชันได้ เนื่องจากเวอร์ชวลแมชชีนเกือบจะไม่ต้องหยุดการทำงาน และทำงานไปด้วยระหว่างที่มีการเซ็คพอยน์ แต่วิธีการนี้มีข้อเสียคือ 1) ในกรณีที่เวอร์ชวลแมชชีนรันแอพพลิเคชันที่มีความต้องการใช้และมีการเขียนหน่วยความจำมาก จะทำให้เวลาในการทำงานของแอพพลิเคชันเพิ่มขึ้นเนื่องจากเวอร์ชวลแมชชีนจะต้องทำงานเซ็คพอยน์ไปด้วย และ 2) เวลาที่ใช้ในการทำเซ็คพอยน์ก็จะเพิ่มขึ้นไปเรื่อยๆตามขนาดของหน่วยความจำที่จะต้องคัดลอก ซึ่งอาจเพิ่มขึ้นจนกระทั่งเวลาที่เซ็คพอยน์สิ้นสุดเมื่อแอพพลิเคชันทำงานเสร็จ ทำให้การเซ็คพอยน์ไม่มีประโยชน์ รวมทั้งขนาดของไฟล์สถานะก็จะใหญ่ขึ้น ทำให้เวลาในการรีสตาร์ทเวอร์ชวลแมชชีนเพิ่มขึ้นด้วย ดังนั้น เพื่อที่จะแก้ไขปัญหานี้และลดโอเวอร์เฮดของเทคนิคไลฟ์ไมเกรชัน งานวิจัยนี้จึงเสนอวิธีการไลฟ์เซ็คพอยน์แบบเทรด (thread-based live checkpointing) ซึ่งหมายถึงการใช้เทรดในไลฟ์ไมเกรชันเพื่อเซ็คพอยน์เวอร์ชวลแมชชีนสำหรับเครื่องโปรเซสเซอร์แบบมัลติคอร์ โดยเทรดจะถูกสร้างขึ้นมาเพื่อช่วยทำการคัดลอกหน่วยความจำแทนเวอร์ชวลแมชชีน โหลดงานของเวอร์ชวลแมชชีนโปรเซสก็จะลดลง ทำให้ประสิทธิภาพการทำงานของแอพพลิเคชันดีขึ้น

วัตถุประสงค์การวิจัย

1. เพื่อออกแบบและสร้างวิธีการใหม่ในการเช็คพอยน์เวอร์ชวลแมชชีนโดยใช้เทคนิคไลฟ์ไมเกรชันแบบเทรด

2. เพื่อศึกษาประสิทธิภาพของวิธีการที่ใช้เทรดในไลฟ์ไมเกรชันสำหรับการเช็คพอยน์เวอร์ชวลแมชชีนและเปรียบเทียบกับประสิทธิภาพของวิธีการอื่น

3. เพื่อศึกษาและเปรียบเทียบขนาดของเช็คพอยน์โอเวอร์เฮดของเทคนิคการเช็คพอยน์เวอร์ชวลแมชชีนด้วยวิธีการไมเกรชันแบบปกติ แบบไลฟ์ไมเกรชัน (บันทึกสถานะลงไฟล์) และแบบไลฟ์ไมเกรชันที่ใช้เทรด

สมมุติฐานการวิจัย

1. เวลาที่ใช้ในการทำงาน (execution time) ของแอพพลิเคชันที่มีการเช็คพอยน์เวอร์ชวลแมชชีนด้วยวิธีการปกติ วิธีการของไลฟ์ไมเกรชัน และวิธีการที่ใช้เทรดในไลฟ์ไมเกรชันมีความแตกต่างกัน

2. เวลาที่ใช้ในการเช็คพอยน์ (checkpoint latency) ระหว่างวิธีการปกติ วิธีการของไลฟ์ไมเกรชัน และวิธีการที่ใช้เทรดในไลฟ์ไมเกรชันมีความแตกต่างกัน

3. การเช็คพอยน์ด้วยวิธีการที่ใช้เทรดในไลฟ์ไมเกรชันมีโอเวอร์เฮด (checkpoint overhead) น้อยกว่าการเช็คพอยน์ด้วยวิธีการอื่น

ขอบเขตของการวิจัย

1. การทดลองในงานวิจัยนี้ทำบนเครื่องโปรเซสเซอร์ 4 คอร์ที่มีประสิทธิภาพสูง (Intel Corporation, 2007, p. 9) เทรดที่ทำเช็คพอยน์จะต้องไม่รันบนคอร์เดียวกับเวอร์ชวลแมชชีน ดังนั้น วิธีการใช้เทรดเพื่อช่วยทำเช็คพอยน์จึงใช้ได้บนเครื่องโปรเซสเซอร์แบบมัลติคอร์เท่านั้น

2. การออกแบบวิธีการที่ใช้เทรดในโปรโตคอลไลฟ์ไมเกรชันเพื่อเช็คพอยน์เวอร์ชวลแมชชีนเป็นการออกแบบที่ใช้ได้ทั่วไป แต่ในงานวิจัยนี้จะใช้การอิมพลีเมนต์บนเควีเอ็ม

ประโยชน์ที่คาดว่าจะได้รับ

1. วิธีการใช้เทรดจะช่วยแก้ปัญหาการเช็คพอยน์ที่ใช้เวลานานของเทคนิคไลฟ์ไมเกรชัน ทำให้สามารถเช็คพอยน์เวอร์ชวลแมชชีนได้เร็วขึ้น

2. แอพพลิเคชันที่เช็คพอยน์โดยใช้วิธีการนี้ทำงานเสร็จเร็วกว่าแอพพลิเคชันที่ใช้วิธีการเช็คพอยน์บนเวอร์ชวลแมชชีนแบบที่มีอยู่ โดยมีเวลาที่หยุดเวอร์ชวลแมชชีนเพียงเล็กน้อย เช่นเดียวกับวิธีการไลฟ์ไมเกรชันปกติ

3. วิธีการใช้เทรดจะช่วยให้เวลาในการเช็คพอยน์ลดลง ซึ่งทำให้สามารถทำเช็คพอยน์ได้มากขึ้น เหมาะสำหรับกรณีแอพพลิเคชันขนาดใหญ่ที่รันเป็นเวลานาน

นิยามศัพท์เฉพาะ

เวอร์ชวลไลเซชัน (virtualization) คือ การสร้างเวอร์ชันเสมือนของอุปกรณ์หรือทรัพยากร เช่น เซิร์ฟเวอร์ อุปกรณ์เก็บข้อมูล เน็ตเวิร์ค หรือระบบปฏิบัติการ

เวอร์ชวลแมชชีน (virtual machine) คือ ระบบปฏิบัติการรวมทั้งแอพพลิเคชันต่างๆที่รันอยู่ในพื้นที่ส่วนหนึ่งที่แยกออกมาต่างหากในเครื่องคอมพิวเตอร์ ซึ่งทำให้ระบบปฏิบัติการที่แตกต่างกันสามารถรันอยู่ภายในคอมพิวเตอร์เครื่องเดียวกันในเวลาเดียวกันได้

เกสโอเอส (guest operating systems) คือ ระบบปฏิบัติการในแต่ละเวอร์ชวลแมชชีนที่สื่อสารกับฮาร์ดแวร์ได้โดยผ่านโปรแกรมควบคุมเวอร์ชวลแมชชีนที่เรียกว่า เวอร์ชวลแมชชีนมอนิเตอร์ ซึ่งทำเวอร์ชวลไลซ์ฮาร์ดแวร์สำหรับแต่ละเวอร์ชวลแมชชีน

เวอร์ชวลแมชชีนมอนิเตอร์ (virtual machine monitor) หรือไฮเปอร์ไวเซอร์ (hypervisor) คือ ซอฟต์แวร์ที่ทำเวอร์ชวลไลเซชัน โดยสร้างเวอร์ชวลแมชชีนขึ้นมาและให้ระบบปฏิบัติการหลายระบบสามารถใช้งานฮาร์ดแวร์ของโฮสร่วมกันได้ จัดการเวอร์ชวลแมชชีนหรือเกสโอเอสต่างๆ โดยให้การสนับสนุนการจัดการเวอร์ชวลไลเซชัน ได้แก่ การจัดสรรทรัพยากรเวอร์ชวลฮาร์ดดิสก์ ไลฟ์ไมเกรชัน

ไลฟ์ไมเกรชัน (live migration) คือ การเคลื่อนย้ายเวอร์ชวลแมชชีนที่กำลังรันอยู่จากโฮสเครื่องหนึ่งไปอีกเครื่องหนึ่งในขณะที่เครื่องยังคงเปิดอยู่ โดยที่ไม่ต้องหยุดการทำงานหรือเซอร์วิสต่างๆที่กำลังรันอยู่ กระบวนการนี้จะเกิดขึ้นโดยที่ไม่มีผลกระทบที่สังเกตเห็นได้จากมุมมองของผู้ใช้

ไลฟ์เช็คพอยน์ติ้ง (live checkpointing) คือ การบันทึกสถานะของเวอร์ชวลแมชชีนลงไฟล์โดยใช้วิธีการของไลฟ์ไมเกรชัน