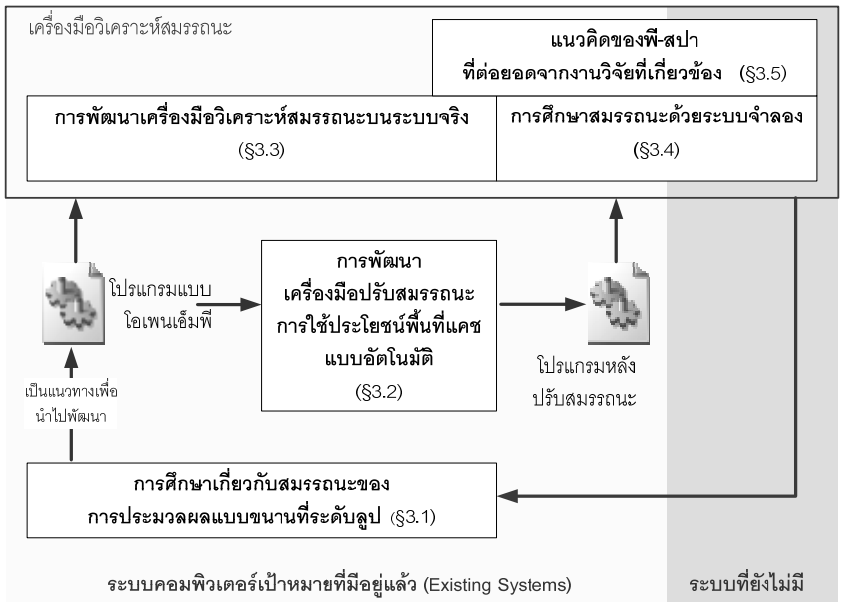
บทที่ 3

งานวิจัยที่เกี่ยวข้อง

วิทยานิพนธ์นี้มีเป้าหมายเพื่อพัฒนาเครื่องมือชื่อพี-สปาเพื่อวิเคราะห์สมรรถนะเกี่ยวกับการใช้ประโยชน์พื้นที่แคช ซึ่งเป็นผลจากการประมวลผลคำสั่งในลูปแบบขนานของโปรแกรมแบบโอเพนเอ็มพี โดยกรอบของแผนงานวิจัยได้ดำเนินการต่อยอดจากงานวิจัยที่เกี่ยวข้องสี่กลุ่มได้แก่ (ภาพที่ 3.1) *งานวิจัยกลุ่มที่หนึ่ง การศึกษาสมรรถนะของการประมวลผลแบบขนานที่ระดับลูป*ของโปรแกรมโอเพนเอ็มพี ซึ่งชี้ว่าผลจากการใช้คอมไพเลอร์สร้างและควบคุมงานเชิงขนาน แม้จะทำให้โปรแกรมพัฒนาได้ง่าย แต่กลับส่งผลให้โปรแกรมเพิ่มสมรรถนะได้จำกัด จึงเกิด*งานวิจัยกลุ่มที่สองซึ่งนำเสนอเครื่องมือปรับการแจกตัวนับลูปอัตโนมัติ* ที่ชี้ให้เห็นว่าสมรรถนะของโปรแกรมสามารถเพิ่มได้ถึงร้อยละ 20 หากปรับรูปแบบการแจกตัวนับให้สอดคล้องกับสถาปัตยกรรมของแคช อย่างไรก็ตามการปรับสมรรถนะอัตโนมัติยังจำกัดเฉพาะโปรแกรมที่มีรูปแบบลูปอย่างปกติเท่านั้น จึงเกิด*งานวิจัยกลุ่มที่สามซึ่งพัฒนาเครื่องวิเคราะห์สมรรถนะบนระบบจริง* ซึ่งทำให้โปรแกรมเมอร์ผู้เชี่ยวชาญสามารถปรับสมรรถนะที่เหมาะกับแต่ละระบบ และ*งานวิจัยกลุ่มสุดท้ายซึ่งนำเสนอระบบจำลองเพื่อใช้วิเคราะห์สมรรถนะของแคช* ที่สามารถวิเคราะห์การเปลี่ยนขนาดของโปรแกรมบนระบบอื่นที่ยังไม่มีจริงได้

ภาพที่ 3.1 ภาพรวมของงานวิจัยที่เกี่ยวข้อง ซึ่งผลักดันทำให้เกิดโครงงานพัฒนาเครื่องมือพี-สปา



การวิจัยเพื่อพัฒนาเครื่องมือพี-สปา รวบรวมแนวคิดดังกล่าวมาพัฒนาต่อยอดเป็นเครื่องมือวิเคราะห์สมรรถนะที่วิเคราะห์เฉพาะการใช้ประโยชน์พื้นที่แคช ซึ่งจะกล่าวถึงรายละเอียดต่อไปในบทนี้

3.1 การศึกษาเกี่ยวกับสมรรถนะของการประมวลผลเชิงขนานที่ระดับลูป

ดังที่กล่าวในบทที่ 1 แล้วว่าโอเพนเอ็มพีมีคำสั่งกำกับคอมไพเลอร์ที่กำหนดวิธีการแบ่งตัวนับลูปไว้สามวิธีให้โปรแกรมเมอร์เลือกใช้ คอมไพเลอร์จะแจกตัวนับลูปไปยังแต่ละเทรดและควบคู่การทำงานระหว่างเทรดโดยอัตโนมัติ โดยที่โปรแกรมเมอร์ไม่จำเป็นต้องกำหนดรายละเอียดการทำงาน อย่างไรก็ตาม คำสั่งกำกับแต่ละคำสั่งมีความเหมาะสมกับคุณลักษณะของโปรแกรม และสถาปัตยกรรมคอมพิวเตอร์แตกต่างกัน ดังนั้น จึงมีการพัฒนาโปรแกรมวัดเปรียบเทียบสมรรถนะหรือเบนช์มาร์ก เพื่อใช้วิเคราะห์สมรรถนะของวิธีการแจกตัวนับลูป และเพื่อหาคำสั่งที่เหมาะสมกับแต่ละแพลตฟอร์ม

3.1.1 การพัฒนาโปรแกรมวัดเปรียบเทียบสมรรถนะ

เนื่องจากคอมไพเลอร์ของโอเพนเอ็มพีจะสร้างและควบคุมการประมวลผลเชิงขนานแบบหลายเทรดโดยอัตโนมัติ ดังนั้น ณ จุดของโปรแกรมที่มีคำสั่งกำกับคอมไพเลอร์จึงเป็นจุดที่อาจจะก่อให้เกิดการสูญเสียสมรรถนะสืบเนื่องจากการสร้างและจัดการเทรด การวัดสมรรถนะของโปรแกรมแบบโอเพนเอ็มพี จึงมักจะประเมินที่การสูญเสียสมรรถนะ โดยจับ *เวลาส่วนเกิน* (Overhead) ที่เกิดขึ้น โดยคำสั่งกำกับที่ใช้เวลาส่วนเกินมากจะมีสมรรถนะต่ำกว่าคำสั่งที่ใช้เวลาส่วนเกินน้อย

โปรแกรมวัดเปรียบเทียบสมรรถนะของคำสั่งกำกับในโปรแกรมแบบโอเพนเอ็มพีชุดแรก คือ อีพีซีซีเบนช์มาร์ก (EPCC) (Bull, 1999) ซึ่งเป็นไมโครเบนช์มาร์กสำหรับวิเคราะห์เวลาส่วนเกินของคำสั่งกำกับคอมไพเลอร์ของโอเพนเอ็มพี เพื่อ (1) เปรียบเทียบสมรรถนะการประมวลผลของคำสั่งกำกับซึ่งถูกสร้างให้เป็นโปรแกรมเชิงขนานแบบหลายเทรด จากคอมไพเลอร์ที่แตกต่างกัน (2) เสนอข้อมูลการสูญเสียสมรรถนะของคำสั่งกำกับแต่ละตัว เพื่อเป็นแนวทางให้โปรแกรมเมอร์เลือกใช้คำสั่งกำกับที่มีสมรรถนะสูงสุด ในกรณีที่คำสั่งกำกับหลายตัวใช้แทนกันได้ และ (3) เป็นแนวทางคาดคะเนเวลาส่วนเกินที่จะขึ้นเมื่อนำโปรแกรมไปประมวลผลจริง ด้วยการคำนวณจากจำนวนคำสั่งกำกับที่ใช้ในโปรแกรม อีพีซีซีแบ่งคำสั่งกำกับออกเป็นสามกลุ่ม

ประกอบด้วย กลุ่มคำสั่งพื้นที่กีดกั้น (Barrier type synchronization) กลุ่มคำสั่งการไม่เกิดร่วม (Mutual exclusion synchronization) และกลุ่มคำสั่งการแจกตัวนับลูป (Loop scheduling)

ผลการใช้อีพีซีซีเบนช์มาร์กเพื่อวิเคราะห์เปรียบเทียบ ระบบคอมพิวเตอร์สามแพลตฟอร์มที่ใช้คอมไพเลอร์ต่างกัน ซึ่งประกอบด้วย คอมไพเลอร์ไคบนเครื่องซันเอชพีซี 3500, มิปส์โปรบนเครื่องเอสจีไอออริจิน 2000 และดิจิทัลบนเครื่องคอมแพกอัลฟ่า พบว่ากลุ่มคำสั่งการแจกตัวนับลูปบนทั้งสามแพลตฟอร์มพบว่า การแบ่งตัวนับลูปด้วยคำสั่ง **`static`**[1] ที่ไม่กำหนดขนาดของบล็อกเป็นวิธีการที่มีการสูญเสียสมรรถนะเป็นเวลาส่วนเกินน้อยที่สุด ขณะที่คำสั่ง **`dynamic(1)`** ทำให้สูญเสียสมรรถนะสูงที่สุด โดยพบว่าการสูญเสียสมรรถนะจากเวลาส่วนเกินนั้น แปรผกผันกับขนาดของบล็อกข้อมูลที่กำหนดในคำสั่ง กล่าวคือ เมื่อกำหนดบล็อกขนาดเล็ก พบว่า คำสั่งกำกับทั้งสามนั้นจะใช้เวลาส่วนเกินสูง แต่เมื่อขยายขนาดของบล็อกให้ใหญ่ขึ้น เวลาส่วนเกินที่ใช้จะลดลง นอกจากนี้ยังพบว่า แพลตฟอร์มที่ใช้คอมไพล์นั้นมีผลต่อเวลาส่วนเกิน ซึ่งพบว่าเครื่องคอมแพกอัลฟ่าที่มีคอมไพเลอร์สำหรับสถาปัตยกรรมคอมแพกเองนั้น เป็นแพลตฟอร์มที่สูญเสียสมรรถนะในแต่ละคำสั่งกำกับน้อยที่สุด

ดิมาโคเปาลอสและคณะ (Dimakopoulos *et al.*, 2008) นำอีพีซีซีเบนช์มาร์กมาพัฒนาเพิ่มเติม เพื่อให้สามารถวิเคราะห์เวลาส่วนเกินที่เกิดจากการประมวลผลแบบขนานที่มีลูปซ้อนกันมากกว่าหนึ่งระดับได้ นักวิจัยศึกษาสมรรถนะของคอมไพเลอร์หกตัวซึ่งมีสามรูปแบบ กล่าวคือ เป็นคอมไพเลอร์ของบริษัทผู้ผลิตเครื่อง (บริษัทซันและอินเทล) คอมไพเลอร์อิสระ (Gnu gcc และ Omni) และคอมไพเลอร์ทดลองของสถาบันการศึกษาผนวกกับไลบรารีจัดการเทรด (OMPi+ POSIX และ OMPi+PS Thread) ผลการทดลองโดยรวมพบว่า บนทุกคอมไพเลอร์นั้น การประมวลผลแบบขนานที่มีลูปซ้อนกันมากกว่าหนึ่งระดับของทั้งสามคำสั่ง จะใช้เวลาส่วนเกินสูงกว่าการประมวลผลแบบขนานบนลูประดับเดียวโดยเฉลี่ยสองเท่า ซึ่งผู้วิจัยได้ตั้งข้อสังเกตว่าคอมไพเลอร์ของโอเพนเอ็มพียังคงมีปัญหาสมรรถนะการประมวลผล หากโปรแกรมเมอร์ต้องการเพิ่มความสามารถในการปรับขนาดได้ของโปรแกรม ด้วยการประมวลผลแบบขนานที่มีลูปซ้อนกันมากกว่าหนึ่งระดับ

โบรเนเวตสกี้และคณะ (Bronevetsky *et al.*, 2008) นำแนวทางของอีพีซีซีเบนช์มาร์กไปพัฒนาโปรแกรมคลอมป์ (CLOMP) เพื่อใช้วิเคราะห์เวลาส่วนเกินบนระบบคอมพิวเตอร์ที่

---

[1] ในส่วนที่เหลือของวิทยานิพนธ์ฉบับนี้ **`static`**, **`dynamic`** และ **`guided`** หมายถึง คำสั่งกำกับ **`static`**, **`dynamic`** และ **`guided`** ที่ไม่กำหนดขนาดของบล็อก ส่วน **`static(n)`**, **`dynamic(n)`** และ **`guided(n)`** หมายถึง คำสั่งกำกับ **`static`**, **`dynamic`** และ **`guided`** ที่กำหนดขนาดของบล็อกเท่ากับ ***n*** ลูป ตามลำดับ

ใช้ในห้องปฏิบัติการแอลแอลเอ็นแอล[2] โปรแกรมคลอมป์ถูกออกแบบให้มีโครงสร้างเหมือนกับแอพพลิเคชันที่ใช้ในห้องปฏิบัติการ ซึ่งมีโครงสร้างลักษณะเฉพาะตัว ทำให้สามารถเลือกใช้คำสั่งกำกับที่เหมาะสมกับแอพพลิเคชันจริงมากกว่าอีพีซีซี โปรแกรมคลอมป์กำหนดให้มีวิธีการแจกตัวนับลูปสามวิธี คือ ใช้คำสั่งกำกับ **`static`**, **`dynamic`** และโปรแกรมเมอร์กำหนดการแจกตัวนับลูปเอง (Manual) ผลการวิเคราะห์บนระบบคอมพิวเตอร์สามแพลตฟอร์ม[3] พบว่า คำสั่ง **`static`** ซึ่งใช้เวลาส่วนเกินน้อยที่สุดจะให้ค่าสปีดอัพหรือประมวลผลเร็วขึ้น 3.9 เท่า ซึ่งใกล้เคียงกับวิธีการแจกตัวนับลูปโดยโปรแกรมเมอร์ที่ประมวลผลเร็วขึ้น 3.4 เท่า ในขณะที่คำสั่ง **`dynamic`** ให้ค่าสปีดอัพ 1.0 เท่า หรือมีสมรรถนะการประมวลผลเร็วคงที่ จากผลการวิเคราะห์นี้แสดงให้เห็นว่า คำสั่ง **`static`** เป็นวิธีการที่เหมาะสมกับแอพพลิเคชั่นจริงในห้องปฏิบัติการแอลแอลเอ็นแอลมากที่สุด

3.1.2 การวิเคราะห์สมรรถนะโดยใช้อีพีซีซีเบนช์มาร์ก

อีพีซีซีเบนช์มาร์กมีขนาดเล็กและติดตั้งง่าย จึงส่งผลให้มีงานวิจัยหลายชิ้นนำไปใช้วิเคราะห์สมรรรถนะบนระบบคอมพิวเตอร์เป้าหมายอื่นเพิ่มเติม เช่นตัวอย่างดังงานวิจัยสามงานต่อไปนี้

เฟรดริกสันและคณะ (Fredrickson *et al*., 2003) ใช้อีพีซีซีวิเคราะห์เวลาส่วนเกินบนเครื่องซันไฟร์ 15 เค ที่มี 288 หน่วยประมวลผล ซึ่งเป็นระบบคอมพิวเตอร์แบบสมมาตรที่มีขนาดใหญ่ (Large-scale symmetric multiprocessors) โดยมีเป้าหมายเพื่อเลือกคำสั่งกำกับที่เหมาะสม เปรียบเทียบสมรรถนะคำสั่งกำกับที่ทดแทนกันได้ และวิเคราะห์ความ สามารถในการปรับขนาดได้ ผลการทดลองพบว่า คำสั่ง **`static`** เป็นคำสั่งที่ทำให้เกิดเวลาส่วนเกินน้อยที่สุด ส่วนคำสั่ง **`dynamic(1)`** เป็นคำสั่งที่ทำให้เกิดเวลาส่วนเกินสูงที่สุด ส่วนการทดลองเพื่อเปรียบเทียบการใช้คำสั่งกำกับแจกตัวนับลูปแบบเพิ่มและไม่เพิ่มเทรด[4] ซึ่งสามารถประมวลผลทดแทนกันได้นั้น พบว่า คำสั่งแบบไม่เพิ่มเทรดซึ่งต้องประกาศภายใต้พื้นที่ส่วนประมวลผลแบบขนานนั้น จะใช้เวลาส่วนเกินสูงกว่าโดยเฉลี่ยร้อยละ 20 ซึ่งชี้ให้เห็นว่า การใช้คำสั่งกำหนดพื้นที่

---

[2] LLNL ย่อมาจาก Lawrence Livermore National Laboratory

[3] Atlas, Thunder และ uP

[4] คำสั่ง **`for`** และ **`do`** ที่ไม่ต้องเพิ่มเทรดนั้น ต้องประกาศภายใต้พื้นที่โปรแกรมที่ประกาศคำสั่ง **`parallel`** ไว้ก่อนล่วงหน้า ในขณะที่คำสั่ง **`parallel for`** และ **`parallel do`** เป็นคำสั่งแจกตัวนับลูปที่จะกำหนดพื้นที่ส่วนประมวลผลแบบขนานหรือสร้างเทรดย่อยขึ้นภายในตัวเองด้วย

ส่วนประมวลผลแบบขนานควบคู่กับคำสั่งการแจกตัวนับลูปรวมเป็นคำสั่งเดียวกัน จะให้สมรรถนะการประมวลผลที่สูงกว่าการใช้คำสั่งทั้งสองชนิดแยกจากกัน สำหรับการวิเคราะห์ความสามารถในการปรับขนาดได้ของโปรแกรมทางวิทยาศาสตร์ 13 โปรแกรม พบว่า แต่ละโปรแกรมมีความสามารถในการปรับขนาดได้แตกต่างกัน ซึ่งขึ้นอยู่กับคุณลักษณะของโปรแกรมและสถาปัตยกรรมของเครื่อง

เลียวและคณะ (Liao *et al*., 2005) นำอีพีซีซีเบนช์มาร์กไปใช้เปรียบเทียบสมรรถนะบนหน่วยประมวลผลแบบหลายเทรดสองสถาปัตยกรรม ระหว่างหน่วยประมวลผลแบบซิปมัลติโพรเซสเซอร์ และหน่วยประมวลผลแบบหลายเทรดต่อเนื่อง หรือ เอสเอ็มที (SMT ย่อมาจาก Simultaneous multithreading) โดยมีเป้าหมายเพื่อเปรียบเทียบเวลาส่วนเกิน และวิเคราะห์ความสามารถในการปรับขนาดได้บนทั้งหน่วยประมวลผลทั้งสองชนิด ผลการทดลองพบว่า คำสั่ง **`static`** เป็นคำสั่งที่ทำให้เกิดเวลาส่วนเกินน้อยที่สุด และคำสั่ง **`dynamic(`*`n`*`)`** เป็นคำสั่งที่ทำให้เกิดเวลาส่วนเกินสูงที่สุด แต่เมื่อนำเวลาส่วนเกินของทั้งสองสถาปัตยกรรมมาเปรียบเทียบกันกลับพบว่า เอสเอ็มทีใช้เวลาส่วนเกินของแต่ละคำสั่งน้อยกว่าซิปมัลติโพรเซสเซอร์ อย่างไรก็ตาม เมื่อประมวลผลชุดโปรแกรมเอ็นพีบี 3.0 และสเปกโอเอ็มพีเอ็ม 2001 เพื่อวิเคราะห์ความสามารถการปรับขนาดได้ปรากฏว่า ซิปมัลติโพรเซสเซอร์ให้ค่าสปีดอัพหรือประมวลผลได้เร็วกว่าเอสเอ็มที เมื่อจำนวนเทรดที่ใช้ประมวลผลเพิ่มขึ้น ผู้วิจัยจึงตั้งข้อสังเกตว่า โปรแกรมแบบโอเพนเอ็มพีเหมาะสมซิปมัลติโพรเซสเซอร์มากกว่าเอสเอ็มที อย่างไรก็ตาม หากพัฒนาให้โปรแกรมแบบโอเพนเอ็มพีมีวิธีการแบ่งตัวนับลูป และสามารถดึงคำสั่งเพื่อเตรียมประมวลผล (Prefetch) ที่เหมาะสมกับสถาปัตยกรรมของเอสเอ็มที ก็จะช่วยเพิ่มสมรรถนะการประมวลผลของโปรแกรมแบบโอเพนเอ็มพีบนเอสเอ็มทีได้

ซูและคณะ (Zhu *et al*., 2006) นำอีพีซีซีเบนช์มาร์กไปทดลองบนระบบจำลองฟาสต์ (FAST) ที่ใช้จำลองการทำงานของหน่วยประมวลผลรุ่นซี 64 ของบริษัทไอบีเอ็ม ที่มีหน่วยประมวลผลย่อยมากถึง 160 หน่วย โดยมีเป้าหมายเพื่อศึกษาเวลาส่วนเกินของคำสั่งกำกับบนหน่วยประมวลผลดังกล่าว ผลการวิเคราะห์พบว่า คำสั่ง **`static`** สูญเสียสมรรถนะต่ำ โดยมีเวลาส่วนเกินน้อยที่สุด ขณะที่คำสั่ง **`dynamic(`*`1`*`)`** จะใช้เวลาส่วนเกินมากที่สุด ทั้งนี้เนื่องจากหากใช้คำสั่ง **`dynamic`** โดยกำหนดบล็อกให้มีขนาดเล็ก จะส่งผลให้โปรแกรมต้องเรียกใช้ฟังก์ชันในการแจกตัวนับลูปบ่อยมากขึ้น และทุกครั้งที่มีการเรียกใช้ฟังก์ชันดังกล่าว ทุกเทรดจะต้องเข้าถึงข้อมูลในตารางเก็บรายละเอียดโพรเซส ที่เรียกว่าคำอธิบายเทรดของเทรดหลักและเทรดย่อย (Master และ Thread descriptor) ซึ่งอยู่บนหน่วยความจำ เพื่อนำมาใช้เป็นข้อมูลในการแบ่งตัวนับลูปให้

แต่ละเทรด ด้วยเหตุนี้ หน่วยประมวลผลซี 64 จึงออกแบบให้แต่ละเทรดจัดเก็บคำอธิบายเทรดไว้บนหน่วยความจำส่วนตัวที่ใกล้ที่สุด เพื่อให้สามารถเข้าถึงข้อมูลดังกล่าวได้เร็วขึ้น

แม้ว่าผลงานวิจัยที่กล่าวมาในข้างต้นจะชี้ให้เห็นว่า คำสั่งกำกับ **dynamic** ซึ่งเป็นวิธีการแจกตัวนับลูปให้หน่วยประมวลผลใดก็ได้ที่พร้อมทำงานเมื่อโปรแกรมรันจะทำให้เกิดเวลาส่วนเกินสูงกว่าคำสั่งอื่น แต่ทว่าการแจกตัวนับลูปด้วยคำสั่งนี้มีข้อดี คือ สามารถช่วยแก้ปัญหาการแจกภาระงานที่ไม่สมดุล (Load imbalance) เนื่องจากการที่ไม่ได้บังคับหมายเลขเทรดที่ต้องรับงานแต่ละบล็อกไว้ล่วงหน้า ทำให้เทรดของหน่วยประมวลผลใดก็ตามที่พร้อมจะรับงานในขณะนั้น รับแจกภาระงานในบล็อกใหม่ที่เหลืออยู่มาประมวลผลได้ทันที โดยไม่จำเป็นต้องรอรับภาระงานแบบเรียงลำดับเหมือนการแจกตัวนับลูปแบบตายตัว อย่างไรก็ตามการแจกงานให้เทรดทำตอนประมวลผล มีข้อจำกัดในกรณีที่ถ้าเทรดนั้นไม่ได้มีข้อมูลพร้อมใช้อยู่แล้ว อาจจะต้องเสียเวลาให้ระบบปฏิบัติการกำหนดค่าข้อมูลเริ่มต้นให้เทรด ดังนั้นสมรรถนะของการแจกตัวนับลูปด้วยคำสั่ง **dynamic** จึงขึ้นอยู่กับคุณลักษณะของแอพพลิเคชันจริงด้วยว่าข้อมูลที่ต้องใช้ในเทรดนั้น มีสำเนาพร้อมใช้ล่วงหน้าหรือไม่

3.1.3 การวิเคราะห์สมรรถนะโดยใช้แอพพลิเคชันจริง

มีตัวอย่างผลงานวิจัยสามชิ้นที่แสดงให้เห็นว่า การใช้คำสั่งแจกลูปแบบ **dynamic** ในแอพพลิเคชันจริงสามารถให้สมรรถนะการประมวลผลสูงกว่าการแจกลูปแบบอื่น ตัวอย่างแรกได้แก่ งานวิจัยของอายกัวเดและคณะ (Ayguadé *et al.*, 2003) ซึ่งเสนอไลบรารี่สำหรับปรับวิธีการแจกลูปอัตโนมัติเพื่อคัดเลือกรูปแบบการแจกตัวนับลูปที่ทำให้โปรแกรมมีค่าสปีดอัพสูงที่สุด การทดลองพบว่า ในโปรแกรมคำนวณค่าผลลัพธ์ของพหุนามเลอร์ชองดร์ (Legendre polynomials) ที่ประมวลผลบนเครื่องพี 690 โดยใช้สี่หน่วยประมวลผล พบว่าการแจกตัวนับลูปด้วยคำสั่ง **dynamic(*16*)** เป็นวิธีที่ให้ค่าสปีดอัพสูงที่สุด เมื่อเปรียบเทียบกับคำสั่ง **static** และ **guided** ตัวอย่างที่สองคือผลงานวิจัยของซางและคณะ (Zhang *et al.*, 2004) ที่แสดงให้เห็นว่าบนโปรแกรมย่อยอาร์ต (art) ของชุดโปรแกรมสเปกโอเพนเอ็มพี วิธีการแจกตัวนับลูปด้วยคำสั่ง **dynamic** ให้ค่าสปีดอัพสูงกว่าคำสั่ง **static** และ **guided** เมื่อประมวลผลบนระบบคอมพิวเตอร์เชิงขนานแบบหน่วยความจำร่วม ที่ใช้หน่วยประมวลผลแบบหลายเทรดต่อเนื่องจำนวน 4 และ 8 หน่วย และตัวอย่างที่สามจากงานวิจัยของเฮอร์เรโรและนาวาร์โร (Herrero & Navarro, 2005) ยังพบว่าวิธีการแจกลูปด้วยคำสั่ง **dynamic** สามารถให้ปริมาณงานได้สูงกว่า

คำสั่ง **`static`** เมื่อประมวลผลโปรแกรมการคูณไฮเปอร์เมทริกซ์ ที่มีขนาดเมทริกซ์ตั้งแต่ 3,680 จนถึง 14,720 ช่อง

จากงานวิจัยที่กล่าวถึงในหัวข้อนี้ แสดงให้เห็นว่าวิธีการแจกตัวนับลูปเพื่อประมวลผลแบบขนานนั้น มีส่วนสำคัญต่อสมรรถนะการประมวลผลของโปรแกรมแบบโอเพนเอ็มพี ทั้งนี้ การสูญเสียสมรรถนะจากการแจกตัวนับลูปแต่ละแบบนั้นจะมากหรือน้อยเท่าใด ขึ้นอยู่กับคุณลักษณะของแอพพลิเคชันจริง แพลตฟอร์มที่ใช้คอมไพล์ รวมถึงสถาปัตยกรรมของระบบคอมพิวเตอร์นั้นด้วย อย่างไรก็ตาม แม้ว่าโอเพนเอ็มพีจะมีวิธีการแจกตัวนับลูปให้เลือกใช้สามวิธี เพื่อให้โปรแกรมเมอร์เลือกใช้วิธีการที่น่าจะให้สมรรถนะการประมวลผลสูงที่สุดแล้ว แต่เนื่องจากคอมไพเลอร์ของโอเพนเอ็มพีจะเป็นผู้สร้างโค้ด เพื่อควบคุมการแจกตัวนับลูปไปยังแต่ละเทรดโดยอัตโนมัติ ทำให้การคัดเลือกหาวิธีการแจกตัวนับลูป เพื่อให้โปรแกรมมีสมรรถนะการประมวลผลสูงขึ้นทำได้จำกัด จึงมีงานวิจัยหลายชิ้นที่นำเสนอเครื่องมือที่ใช้ทำงานร่วมกับคอมไพเลอร์ของโอเพนเอ็มพีโดยอัตโนมัติ ซึ่งเครื่องมือเหล่านี้จะช่วยเพิ่มสมรรถนะการประมวล ผลแบบขนานที่ระดับลูป ด้วยการเพิ่มประสิทธิภาพการใช้ประโยชน์พื้นที่แคชของโปรแกรม โดยจะได้กล่าวถึงรายละเอียดในหัวข้อถัดไป

3.2 การพัฒนาเครื่องมือปรับสมรรถนะการใช้ประโยชน์พื้นที่แคชแบบอัตโนมัติ

แม้ว่าโอเพนเอ็มพีได้เตรียมคำสั่งกำกับที่ใช้ในการแจกตัวนับลูปไว้ให้โปรแกรมเมอร์เลือกใช้ถึงสามคำสั่ง พร้อมทั้งอนุญาตให้กำหนดขนาดของบล็อกได้เอง เพื่อความยืดหยุ่นในการแจกตัวนับลูป แต่อย่างไรก็ตาม การที่โปรแกรมแบบโอเพนเอ็มพีใช้คอมไพเลอร์สร้างโค้ดเพื่อประมวลผลแบบขนานโดยอัตโนมัตินั้น ทำให้โปรแกรมเมอร์ปรับความสมดุลการแจกตัวนับลูปได้จำกัด ถึงแม้ว่าโปรแกรมเมอร์ผู้เชี่ยวชาญอาจเลือกใช้คำสั่งการแจกตัวนับลูป ที่เหมาะสมกับสถาปัตยกรรมของเครื่องได้ ทำให้โปรแกรมสามารถใช้ทรัพยากรของเครื่อง เช่น การใช้ประโยชน์พื้นที่แคชได้อย่างเต็มที่ แต่ในขณะเดียวกัน โปรแกรมเมอร์ทั่วไปอาจเลือกใช้ข้อกำหนดที่ไม่เหมาะสม ทำให้โปรแกรมสูญเสียสมรรถนะการประมวลผลได้เช่นกัน จึงมีงานวิจัยหลายชิ้นที่นำเสนอเครื่องมือช่วยปรับวิธีการแจกตัวนับลูป บนโปรแกรมเชิงขนานให้เหมาะสมกับสถาปัตยกรรมของเครื่องโดยอัตโนมัติ ซึ่งเครื่องมือเหล่านี้จะช่วยให้โปรแกรมสามารถแจกตัวนับลูปได้อย่างสมดุล และสามารถใช้ประโยชน์พื้นที่แคชได้อย่างเต็มประสิทธิภาพ

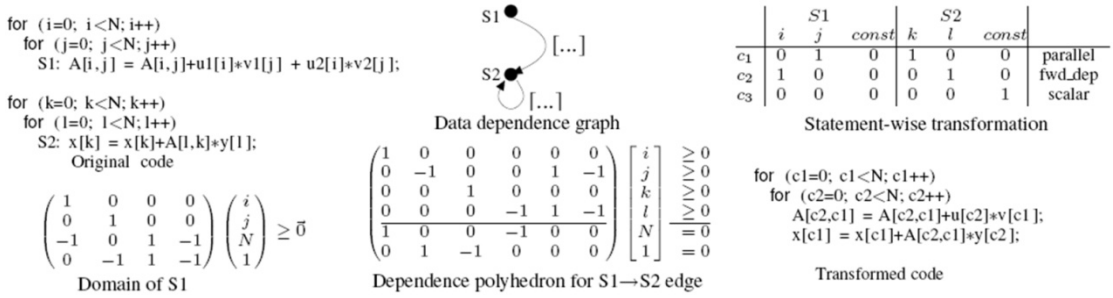
3.2.1 เครื่องมือปรับสมรรถนะแบบไลบราลีช่วยประมวลผล

อายกัวเดและคณะ (Ayguadé *et al*., 2003) ได้นำเสนอไลบราลีช่วยการแจกตัวนับลูประหว่างประมวลผล บนโปรแกรมแบบโอเพนเอ็มพี ที่สามารถปรับวิธีการแจกลูปให้เหมาะสมกับการประมวลผลในแต่ละลูปได้โดยอัตโนมัติ โดยไลบราลีดังกล่าวนี้จะปรับใช้วิธีการแจกตัวนับลูปที่สามารถเพิ่มคุณสมบัติพื้นที่ที่ถูกใช้บ่อยครั้งและพื้นที่ซึ่งถูกใช้บริเวณติดกัน เพื่อลดอัตราการไม่พบข้อมูลบนแคช และสามารถช่วยกระจายภาระงานในลูปได้อย่างสมดุล ผลการทดสอบไลบราลีดังกล่าวนี้ บนเครื่องพี 690 ที่ใช้หน่วยประมวลผลไอบีเอ็ม พาวเวอร์ 4 จากการประมวลผลชุดโปรแกรมเอ็นพีบี พบว่า โปรแกรมที่ใช้ไลบราลีช่วยในการแจกลูปให้ค่าสปีดอัพไม่แตกต่างกับโปรแกรมต้นฉบับ เนื่องจาก การประมวลผลในลูปของโปรแกรมเอ็นพีบีได้เลือกใช้วิธีการแจกตัวนับลูปที่ทำให้มีอัตราการไม่พบข้อมูลบนแคชต่ำ และสามารถแจกตัวนับลูปได้อย่างสมดุลอยู่แล้ว ส่วนการประมวลผลบนโปรแกรมย่อยแอมเอ็มพี (ammp) ของชุดโปรแกรมสเปก พบว่าโปรแกรมที่ใช้ไลบราลีช่วยในการแจกตัวนับลูป สามารถช่วยให้อัตราการไม่พบข้อมูลบนแคชลดลง ส่งผลให้ค่าสปีดอัพหรือมีสมรรถนะเพิ่มขึ้นจากโปรแกรมต้นฉบับร้อยละ 27 และ 22 บนหน่วยประมวลผล 4 และ 16 หน่วย ตามลำดับ

จากแนวทางการพัฒนาไลบราลีเพื่อช่วยในการแจกตัวนับลูปประมวลผล ซางและคณะ (Zhang *et al*., 2004) ได้นำเสนอไลบราลีช่วยในการแจกตัวนับลูปเพื่อใช้บนหน่วยประมวลผลแบบเอสเอ็มที ซึ่งไลบราลีดังกล่าวนี้สามารถปรับจำนวนเทรดให้เหมาะสมกับการประมวลผลในแต่ละลูปได้ด้วย เนื่องจากการประมวลผลของเอสเอ็มที ทุกเทรดต้องใช้ทรัพยากรบนหน่วยประมวลผล เช่น แคช ร่วมกัน จำนวนเทรดของแต่ละหน่วยประมวลผลจึงเป็นปัจจัยสำคัญที่ส่งผลต่อสมรรถนะการประมวลของโปรแกรมด้วย จากการประมวลผลชุดโปรแกรมสเปกบนเครื่องให้บริการ (Server) ที่มี 4 หน่วยประมวลผล พบว่า โปรแกรมที่ใช้ไลบราลีช่วยในการแจกลูปให้ค่าสปีดอัพสูงกว่าโปรแกรมต้นฉบับเฉลี่ยร้อยละ 10 และร้อยละ 20 บนการประมวลผล 4 และ 8 เทรดตามลำดับ ผลงานวิจัยของซางแสดงให้เห็นว่าการแจกตัวนับลูปที่อยู่ต่อเนื่องกันไปยังเทรดที่มีลำดับติดกัน และประมวลผลอยู่บนหน่วยประมวลผลเดียวกันนั้น จะทำให้พื้นที่เก็บข้อมูลในแคชถูกเข้าถึงในลักษณะบริเวณติดกันเพิ่มมากขึ้น ซึ่งจะทำให้อัตราการไม่พบข้อมูลบนแคชลดลง เนื่องจาก โดยส่วนใหญ่ข้อมูลที่ใช้ในการประมวลผลในลูปนั้นจะถูกเก็บไว้เป็นโครงสร้างของอาร์เรย์ หากอาร์เรย์ที่อยู่ติดกันถูกดึงเข้ามาบนแคชของหน่วยประมวลผลเดียวกัน จะทำให้สามารถดึงข้อมูลเข้าไปประมวลผลได้อย่างต่อเนื่องและเร็วขึ้น

3.2.2 เครื่องมือปรับสมรรถนะแบบคอมไพเลอร์

จากคุณสมบัติการจัดเก็บข้อมูลในลูปประมวลผลด้วยโครงสร้างของอาร์เรย์นี้ งานวิจัยของอิชิซาเกะและคณะ (Ishizaka *et al*., 2004) ได้นำเสนอวิธีการแบ่งและจัดเรียงข้อมูลในอาร์เรย์ให้เหมาะสมกับสถาปัตยกรรมของแคช และแบ่งตัวนับลูปที่ต่อเนื่องกันไปยังหน่วยประมวลผลเดียวกัน เพื่อช่วยลดการไม่พบข้อมูลที่ขัดแย้งกัน และช่วยเพิ่มประสิทธิภาพการใช้ประโยชน์พื้นที่แคช อิชิซาเกะนำวิธีการนี้ไปพัฒนาเพิ่มเติมบนคอมไพเลอร์ออสการ์ (OSCAR) และนำไปทดสอบบนเครื่องซัน อัลตรา 80 และเครื่องไอบีเอ็ม อาร์เอส/6000 ซึ่งทั้งสองเครื่องมีสถาปัตยกรรมของแคชระดับที่ 2 ต่างกัน จากการประมวลผลชุดโปรแกรมสเปกบนเครื่องซันอัลตรา 80 พบว่า คอมไพเลอร์ออสการ์ที่ได้รับการปรับปรุงสามารถช่วยให้การไม่พบข้อมูลในแคชระดับที่ 2 ลดลงเฉลี่ยร้อยละ 22 ซึ่งส่งผลให้โปรแกรมมีค่าสปีดอัพเพิ่มขึ้นเฉลี่ย 3.5 เท่า จากคอมไพเลอร์แบบดั้งเดิม ส่วนบนเครื่องไอบีเอ็ม อาร์เอส/6000 พบว่าโปรแกรมที่ถูกคอมไพล์ใหม่มีค่าสปีดอัพเพิ่มขึ้นเฉลี่ย 2.5 เท่า ผลการทดลองชี้ให้เห็นว่า วิธีการแบ่งและจัดเรียงอาร์เรย์ของข้อมูลในลูปประมวลผลที่เหมาะสมกับสถาปัตยกรรมของแคช จะช่วยให้อัตราการไม่พบข้อมูลลดลงได้ ซึ่งงานวิจัยของอิชิซาเกะนำวิธีการดังกล่าวมาพัฒนาเพิ่มเติมบนคอมไพเลอร์ ทำให้สามารถแบ่งและจัดเรียงข้อมูลได้โดยอัตโนมัติขณะที่คอมไพล์โปรแกรม



(a) การตรวจสอบความเป็นอิสระระหว่างข้อมูล (b) การแมปโครงสร้างลูปและข้อมูลให้อยู่ในรูปแบบโพลีฮีดรัล (c) การสร้างโค้ดลูปประมวลผลใหม่โดยอัตโนมัติ

ภาพที่ 3.2 ขั้นตอนการปรับโครงสร้างลูปแบบโพลีฮีดรัล (a) การตรวจสอบความเกี่ยวข้องกันของข้อมูล (b) การแมปโครงสร้างลูปและข้อมูลให้อยู่ในรูปแบบโพลีฮีดรัล และ (c) การสร้างโค้ดลูปประมวลผลใหม่โดยอัตโนมัติ (Bondhugula *et al*., 2008)

จากแนวทางการพัฒนาคอมไพเลอร์ เพื่อให้สามารถเพิ่มประสิทธิภาพการใช้ประโยชน์พื้นที่แคชนี้ บอนดูห์กูลาและคณะ (Bondhugula *et al*., 2008) ได้นำเสนอวิธีการปรับโครงสร้างลูปแบบโพลีฮีดรัล (Polyhedral model) สำหรับลูปประมวลผลที่ซ้อนกันมากกว่าหนึ่งระดับ โดยนำไปพัฒนาบนคอมไพเลอร์พลูโต (PLuTo) ซึ่งเป็นคอมไพเลอร์แบบก่อนคอมไพล์จริง

(Pre-processor) สำหรับโปรแกรมแบบโอเพนเอ็มพี เป้าหมายของการปรับโครงสร้างลูปแบบโพลีฮีดรัลนั้น เพื่อช่วยเพิ่มสัดส่วนการประมวลผลแบบขนานที่ระดับลูปให้สูงขึ้นแบบอัตโนมัติ และเพิ่มประสิทธิภาพการเข้าถึงพื้นที่เก็บข้อมูลได้ดียิ่งขึ้น การทำงานของการปรับโครงสร้างลูปแบบโพลีฮีดรัล สามารถแบ่งออกเป็นสามขั้นตอนหลัก (ภาพที่ 3.2) ประกอบด้วย (1) การตรวจสอบความเกี่ยวข้องกันของข้อมูลในที่ใช้ประมวลผลในลูป (2) การแมปโครงสร้างลูปประมวลผลและข้อมูลให้อยู่ในรูปแบบโพลีฮีดรัล และ (3) การสร้างโค้ดลูปประมวลผลใหม่โดยอัตโนมัติ

บอนดูห์กูลานำพลูโตไปทดสอบเพื่อวิเคราะห์ค่าสปีดอัพที่เพิ่มขึ้น บนโปรแกรมการคำนวณทางวิทยาศาสตร์ห้าโปรแกรม[5] พบว่า โปรแกรมทั้งหมดเมื่อถูกคอมไพล์ด้วยคอมไพเลอร์พลูโตแล้ว สามารถให้ค่าสปีดอัพเพิ่มขึ้น 2 ถึง 5 เท่า เมื่อเทียบกับคอมไพเลอร์อื่น ซึ่งผลการทดลองแสดงให้เห็นว่า การปรับโครงสร้างลูปประมวลผลให้มีสัดส่วนการประมวลผลแบบขนานเพิ่มขึ้น จะช่วยให้โปรแกรมมีความสามารถในการปรับขนาดได้สูงขึ้น และการปรับโครงสร้างข้อมูลที่ถูกใช้ประมวลผลในลูป สามารถช่วยทำให้การเข้าถึงข้อมูลของหน่วยประมวลผลทำได้เร็วขึ้นด้วย

เครื่องมือปรับสมรรถนะโปรแกรมแบบโอเพนเอ็มพีโดยเพิ่มการใช้ประโยชน์พื้นที่แคชที่กล่าวมามักจะพัฒนาเป็นไลบรารีหรือไม่ก็ผนวกเข้ากับคอมไพเลอร์ จึงสามารถปรับสมรรถนะของโปรแกรมได้โดยอัตโนมัติ อย่างไรก็ตาม จากผลงานวิจัยที่ผ่านมาอาจตั้งข้อสังเกตได้สองประการ คือ (1) การเพิ่มสมรรถนะของโปรแกรมนั้นจำกัดเพียงบางส่วนโปรแกรมส่วนใหญ่มีสมรรถนะเฉลี่ยสูงขึ้นเพียงร้อยละ 20 และ (2) หากแอพพลิเคชันจริงมีรูปแบบโครงสร้างลูปและการเข้าถึงข้อมูลที่ไม่ปกติ เครื่องมือที่กล่าวมาอาจจะไม่สามารถช่วยเพิ่มสมรรถนะได้ ด้วยเหตุนี้จึงมีการพัฒนาเครื่องมือวิเคราะห์สมรรถนะเพื่อใช้วิเคราะห์จุดบอดเชิงสมรรถนะของโปรแกรมและปรับปรุงสมรรถนะได้ตามรูปแบบเฉพาะของโปรแกรม ซึ่งเครื่องมือวิเคราะห์สมรรถนะเหล่านี้มีรูปแบบการทำงานสองลักษณะดังรายละเอียดในหัวข้อถัดไป

[5] Jacobi stencil, Finite Difference Time Domain, LU decomposition, Matrix vector transpose และ 3-D Gauss-Seidel

ตารางที่ 3.1 คุณลักษณะของเครื่องมือวิเคราะห์สมรรถนะโปรแกรมแบบโอเพนเอ็มพี

คุณลักษณะ	เครื่องมือวิเคราะห์สมรรถนะ							
	การวัดค่าบนระบบจริง					ระบบจำลอง		
	ดรากอน	ออมพี	ซัน สตูดิโอ	โคแจก	ทาว	ดีซิม คลัสเตอร์	ซิมที/ โอเอ็มพี	ซีซีซิม
ความสามารถในการรองรับ								
รูปแบบการเขียนโปรแกรม								
โอเพนเอ็มพี	√	√	√	√	√	√	√	√
เอ็มพีไอ	–	–	√	√	√	–	–	–
(พีเทรด, โพซิกซ์)	–	–	√	√	√	–	–	–
ภาษาโปรแกรม								
ซี / ซีพลัสพลัส	√	√	√	√	√	√	√	√
ฟอร์แทรน	√	√	√	√	√	–	–	–
(จาวา, ไพธอน)	–	–	√	–	√	–	–	–
สถาปัตยกรรม								
x86 และ x86_64	–	√	√	√	√	–	√	√
IA-64	√	√	–	√	√	–	–	–
Sun SPARC	–	–	√	√	√	√	–	–
PowerPC	–	–	–	√	√	–	–	–
ระบบปฏิบัติการ								
Unix และ Linux	√	√	√	√	√	√	√	√
Solaris	–	√	√	√	√	–	–	–
Windows	–	–	–	√	√	–	–	–
ข้อมูลเชิงสมรรถนะ								
หน่วยวัดสมรรถนะ								
เวลาประมวลผล	√	√	√	√	√	√	√	√
เวลาส่วนเกิน	–	√	–	–	–	–	–	–
จำนวนการเรียกใช้ฟังก์ชัน	√	–	√	√	√	–	√	√
การใช้ประโยชน์พื้นที่แคช								
อัตราการไม่พบข้อมูล	–	–	√	√	√	√	√	√
ชนิดการไม่พบข้อมูล	–	–	–	–	–	–	–	√
คุณสมบัติการเข้าถึงข้อมูล	–	–	–	–	–	√	√	–
รูปแบบการนำเสนอข้อมูล								
ไฟล์ข้อความ	√	√	–	–	√	√	√	√
กราฟ	√	–	√	√	√	√	√	√
ไดอะแกรมเชิงเวลา	–	–	√	√	√	–	–	–
แผนภาพการเรียกใช้ฟังก์ชัน	√	–	√	√	√	–	√	–
ภาพเคลื่อนไหว	–	–	–	–	–	√	–	–

3.3 การเพิ่มสมรรถนะของโปรแกรมแบบโอเพนเอ็มพีด้วยเครื่องมือวิเคราะห์สมรรถนะ

การพัฒนาโปรแกรมเชิงขนานเพื่อให้ได้สมรรถนะการประมวลผลที่สูงนั้น โปรแกรมต้องสามารถเข้าใช้ทรัพยากรของเครื่องได้อย่างคุ้มค่า แต่ทว่าจากรายงานของดอนการ์รา (Dongarra, 2008) แสดงให้เห็นว่า การประมวลผลโปรแกรมแก้สมการเชิงเส้นลินแพค (LINPACK) บนระบบคอมพิวเตอร์มากกว่า 500 แพลตฟอร์มให้สมรรถนะการประมวลผลได้มีเฉลี่ยเพียงร้อยละ 50 ของสมรรถนะการประมวลผลสูงสุดตามหลักทฤษฎีเท่านั้น การวิเคราะห์หาสาเหตุหรือปัจจัยที่เป็นข้อจำกัดเชิงสมรรถนะ จึงเป็นประเด็นที่ท้าทายและได้รับความสนใจจากนักวิจัย ทำให้มีการพัฒนาเครื่องมือวิเคราะห์สมรรถนะที่ช่วยนำเสนอข้อมูลเพื่อให้โปรแกรมเมอร์นำไปปรับปรุงโปรแกรม ให้สามารถเข้าใช้ทรัพยากรของเครื่องได้อย่างเต็มประสิทธิภาพ

ในช่วงทศวรรษที่ผ่านมา มีงานวิจัยหลายชิ้นที่นำเสนอเครื่องมือวิเคราะห์สมรรถนะสำหรับโปรแกรมเชิงขนาน ซึ่งอาจแบ่งตามลักษณะการทำงานได้เป็นสองประเภท (Leko *et al.*, 2005) คือ (1) เครื่องมือแบบคัดลักษณะเฉพาะหรือโพรไฟลิ่ง (Profiling) ที่ใช้วิธีการสุ่มเก็บตัวอย่างข้อมูลขณะที่โปรแกรมกำลังประมวลผล เพื่อนำมาสรุปเป็นข้อมูลเชิงสมรรถนะ เช่น เวลาประมวลผลของโปรแกรม โดยไม่เก็บรายละเอียดขั้นตอนการประมวลผลทั้งหมดของโปรแกรม และ (2) เครื่องมือแบบแกะรอยหรือเทรซซิง (Tracing) ที่ใช้วิธีบันทึกลำดับเหตุการณ์ที่เกิดขึ้นระหว่างการประมวลผล ลงบนไฟล์เก็บขั้นตอนการทำงาน (Trace file) เพื่อนำไฟล์ดังกล่าวมาจำลองการประมวลผลของโปรแกรมในภายหลัง ตารางที่ 3.1 แสดงรายการเครื่องมือวิเคราะห์สมรรถนะทั้งสองประเภท รวมทั้งสิ้นแปดเครื่องมือ และคุณลักษณะเฉพาะของแต่ละเครื่องมือเปรียบเทียบกัน ดังรายละเอียดที่จะกล่าวถึงในหัวข้อต่อไปนี้

3.3.1 เครื่องมือวิเคราะห์สมรรถนะแบบโพรไฟลิ่ง

เครื่องมือวิเคราะห์สมรรถนะแบบโพรไฟลิ่ง ซึ่งประกอบด้วย ดรากอน ออมพี และซันสตูดิโอ ซึ่งมีรายละเอียดดังต่อไปนี้

3.3.1.1 ดรากอน (Dragon)

ดรากอน (Hernandez *et al.*, 2005) ถูกพัฒนาโดยนักวิจัยของมหาวิทยาลัยฮูสตัน เพื่อเป็นเครื่องมือสำหรับวิเคราะห์สมรรถนะโปรแกรมแบบโอเพนเอ็มพี บนหน่วยประมวลผลที่ใช้ชุดคำสั่งสถาปัตยกรรมไอเอ-64 (IA-64) ของบริษัทอินเทล โดยสามารถรองรับโปรแกรมที่พัฒนา

ด้วยภาษาซี/ซีพลัสพลัส และฟอร์แทรน ดรากอนวิเคราะห์โปรแกรมภาษาระดับสูงแล้วจึงแสดงค่าสมรรถนะและผลการวิเคราะห์ของโปรแกรมในสี่รูปแบบ ได้แก่ (1) เวลาประมวลผล (2) จำนวนการเรียกใช้ฟังก์ชัน (3) ลักษณะความขึ้นต่อกันของข้อมูล (Data dependency) ภายในลูป ซึ่งจำแนกเป็น การรออ่านข้อมูลซึ่งเขียนโดยคำสั่งอื่น (True dependency), การรอเขียนข้อมูลหลังจากที่คำสั่งอื่นอ่าน (Anti dependency), การเขียนข้อมูลหลังจากที่คำสั่งอื่นเขียนก่อน (Output dependency) และ (4) ความเกี่ยวข้องระหว่างกระบวนงาน (Procedure dependency) ข้อมูลเชิงสมรรถนะเหล่านี้จะถูกเก็บเป็นไฟล์ข้อความ กราฟความเกี่ยวข้องของข้อมูล และแผนภาพการเรียกใช้ฟังก์ชัน (Call graph)

ดรากอนได้ถูกนำไปใช้วิเคราะห์สมรรถนะของโปรแกรมทางวิทยาศาสตร์ ซึ่งเป็นโปรแกรมคำนวณการไหลของสารเคมี บนชุดโปรแกรมวัดสมรรถนะเอ็นพีบีขององค์การนาซ่า ผลการวิเคราะห์ด้วยแผนภาพการเรียกใช้ฟังก์ชันพบว่า โปรแกรมดังกล่าวมีฟังก์ชันที่ใช้ประมวลผลหลักสามฟังก์ชัน ซึ่งทั้งสามฟังก์ชันเป็นการประมวลผลแบบขนานที่ระดับลูปซ้อนกันสามระดับ และเมื่อวิเคราะห์ความเกี่ยวของข้อมูลภายในลูป พบว่า คำสั่งภายในลูปชั้นในสุดมีความเกี่ยวข้องกันในลักษณะที่มีการรออ่านข้อมูลซึ่งเขียนโดยคำสั่งอื่น ทำให้ไม่สามารถแจกลูปเพื่อประมวลผลแบบขนานที่ระดับนี้ได้ ดังนั้น หากต้องการเพิ่มสมรรถนะของโปรแกรม โปรแกรมเมอร์จะต้องปรับวิธีการแจกตัวนับลูปที่ลูปประมวลผลสองชั้นด้านนอกเท่านั้น จากผลการทดลองแสดงให้เห็นว่าแผนภาพการเรียกใช้ฟังก์ชันที่ดรากอนนำเสนอ สามารถช่วยวิเคราะห์ส่วนประมวลผลหลักของโปรแกรมได้ เพื่อให้โปรแกรมเมอร์สามารถเลือกปรับปรุงเฉพาะบางส่วนของโปรแกรม โดยไม่จำเป็นต้องปรับปรุงโปรแกรมทั้งหมด

3.3.1.2 **ออมพี** (ompP)

ออมพี (Fürlinger & Gerndt, 2005) ถูกพัฒนาโดยนักวิจัยจากมหาวิทยาลัยเทนเนสซี เพื่อใช้เป็นเครื่องมือวิเคราะห์สมรรถนะสำหรับโปรแกรมแบบโอเพนเอ็มพีโดยเฉพาะ ซึ่งสามารถรองรับโปรแกรมแบบโอเพนเอ็มพีได้ทุกภาษา และทำงานบนระบบปฏิบัติการตระกูลยูนิกซ์ได้ทั้งหมด ออมพีมีเป้าหมายเพื่อวิเคราะห์ถึงความสมดุลของการกระจายภาระงาน และเวลาส่วนเกินที่ใช้ทำงานร่วมกันระหว่างเทรด โดยจะแยกวิเคราะห์พื้นที่ส่วนประมวลแบบขนานออกเป็นส่วนย่อย ตามที่ระบุไว้ด้วยคำสั่งกำกับคอมไพเลอร์ จากภาพที่ 3.3 แสดงถึงชนิดของพื้นที่ส่วนย่อยที่เกิดจากคำสั่งกำกับ ซึ่งแต่ละพื้นที่จะมีคุณสมบัติของการทำงานร่วมกันระหว่างเทรดที่แตกต่างกัน เช่น `enter` เป็นเวลาที่แต่ละเทรดใช้เข้าสู่พื้นที่ส่วนประมวลผลแต่ละส่วน ขณะที่

exit เป็นเวลาที่ใช้ออกจากพื้นที่ส่วนประมวลผล ออมพีจะนำเสนอในรูปแบบไฟล์ข้อความ ที่แสดงเวลาการประมวลผลของแต่ละเทรดบนแต่ละพื้นที่ส่วนประมวลผลแบบขนาน ซึ่งพื้นที่แต่ละส่วนจะระบุลำดับแถวของโค้ดในโปรแกรมไว้ด้วย ทำให้โปรแกรมเมอร์ค้นหาพื้นที่ดังกล่าวเร็วขึ้น

ภาพที่ 3.3 ชนิดของพื้นที่ส่วนย่อยที่เกิดจากคำสั่งกำกับคอมไพเลอร์และคุณสมบัติของแต่ละพื้นที่ ซึ่งถูกแบ่งโดยเครื่องมือวิเคราะห์สมรรถนะออมพี (Fürlinger & Gerndt, 2005)

	seq	main	body	ibarr	enter	exit
MASTER	×					
ATOMIC		×				
BARRIER		×				
FLUSH		×				
USER_REGION		×				
LOOP		×		×		
SECTIONS		×	×	×		
SINGLE		×	×	×		
CRITICAL		×	×		×	×
WORKSHARE		×		×		
PARALLEL	×		×	×	×	×
PARALLEL_LOOP	×		×	×	×	×
PARALLEL_SECTIONS	×	×	×	×	×	×
PARALLEL_WORKSHARE	×		×	×	×	×

ในงานวิจัยของเฟอร์ลิงเกอร์ซึ้นนี้ ได้นำเสนอการใช้ออมพีวิเคราะห์เพื่อหาส่วนของโปรแกรมที่กระจายภาระงานไม่สมดุล บนชุดโปรแกรมเอทีเอส (ATS ย่อมาจาก APART test suite) และโปรแกรมควิกซอร์สแบบขนาน พบว่า ออมพีสามารถตรวจพบส่วนของโปรแกรมที่มีปัญหาการกระจายภาระงานที่ไม่สมดุลบนทั้งสองโปรแกรมได้ สำหรับบนโปรแกรมควิกซอร์สแบบขนานนั้น เมื่อนำข้อมูลที่ได้ไปปรับปรุงทำให้โปรแกรมประมวลผลได้เร็วขึ้นประมาณร้อยละ 25 ของโปรแกรมเดิม จากผลการทดลองชี้ให้เห็นว่าปัญหาการกระจายภาระงานที่ไม่สมดุลจะทำให้โปรแกรมสูญเสียสมรรถนะการประมวลผล ดังนั้น เครื่องมือที่สามารถตรวจพบปัญหาดังกล่าวนี้ได้จะช่วยให้การพัฒนาโปรแกรมแบบโอเพนเอ็มพีที่มีสมรรถนะสูงทำได้เร็วขึ้น

3.3.1.3 **ซันสตูดิโอ** (Sun Studio Performance Analyzer)

เครื่องมือวิเคราะห์สมรรถนะซันสตูดิโอ (Jost *et al*., 2006) ถูกพัฒนาโดยบริษัทซัน เพื่อใช้เป็นเครื่องมือสำหรับใช้วิเคราะห์โปรแกรมทั้งแบบลำดับและแบบขนานที่พัฒนาด้วยโพซิกซ์ (Posix) เอ็มพีไอ และโอเพนเอ็มพี บนระบบปฏิบัติการโซลาริส (Solaris) และลินิกซ์ (Linux) สำหรับการวิเคราะห์สมรรถนะโปรแกรมแบบโอเพนเอ็มพีนั้น ซันสตูดิโอได้พัฒนาไลบรารีเพิ่มเติมเพื่อใช้เก็บสถานะการประมวลผลของแต่ละเทรด เพื่อให้ทราบถึง เวลาประมวลผล เวลาเดินเครื่อง

เปล่า และความสมดุลของการกระจายภาระงานของแต่ละเทรด ทั้งนี้ซันสตูดิโอสามารถนำเสนอข้อมูลได้หลายรูปแบบ เช่น กราฟ ไดอะแกรมเชิงเวลา และแผนภาพการเรียกใช้ฟังก์ชัน

โจสต์และคณะได้นำเสนอการใช้ซันสตูดิโอวิเคราะห์โปรแกรมอากาศพลศาสตร์ ทีเอฟเอส (TFS) เพื่อวิเคราะห์ความสมดุลของการกระจายภาระงาน และอัตราการเข้าถึงข้อมูลข้ามโหนด (Remote access) เมื่อประมวลผลบนบนเครื่องซันไฟร์ อี6900 และซันไฟร์ 25 เค ซึ่งเป็นระบบคอมพิวเตอร์แบบซีซี-นูมา (cc-NUMA) ผลการวิเคราะห์พบว่า โปรแกรมทีเอฟเอสให้ค่าสปีดอัพได้เพียง 5 เท่า ซึ่งสาเหตุที่ทำให้โปรแกรมมีสมรรถนะการประมวลผลต่ำนั้น เนื่องจากมีส่วนของโปรแกรมที่มีปัญหาการกระจายงานไม่สมดุล จึงทำให้เกิดเวลาเดินเครื่องเปล่าของแต่ละเทรดเกินกว่าร้อยละ 65 ของเวลาประมวลผลทั้งหมด และมีอัตราการเข้าถึงข้อมูลข้ามโหนดเกิดขึ้นถึงร้อยละ 23 ของการเข้าถึงข้อมูลทั้งหมดของโปรแกรม ผลการวิเคราะห์ให้ผลเช่นเดียวกับงานของเฟอร์ลิงเกอร์ที่กล่าวถึงข้างต้น กล่าวคือ การกระจายภาระงานที่ไม่สมดุลทำให้โปรแกรมสูญเสียสมรรถนะการประมวลผล และมีความสามารถในการปรับขนาดได้ต่ำ

3.3.2 เครื่องมือวิเคราะห์สมรรถนะแบบเทรซซิง

ในหัวข้อนี้กล่าวถึง เครื่องมือวิเคราะห์สมรรถนะแบบเทรซซิง ซึ่งประกอบด้วย โคแจก และทาว ซึ่งมีรายละเอียดดังต่อไปนี้

3.3.2.1 โคแจก (KOJAK[6])

โคแจก (Mohr & Wolf, 2003) ถูกพัฒนาโดยสถาบัน Forschungszentrum Jülich เพื่อใช้วิเคราะห์โปรแกรมแบบโอเพนเอ็มพี เอ็มพีไอ และพีเทรด (Pthread) บนภาษาซี ซีพลัสพลัส และฟอร์แทรน โดยมีเป้าหมายของการวิเคราะห์เพื่อหาส่วนของโปรแกรมที่ทำให้เกิดปัญหาคอขวด โคแจกจะนำเสนอข้อมูลซึ่งแบ่งออกเป็นสามลักษณะ คือ (1) ข้อมูลคุณสมบัติเชิงสมรรถนะของแต่ละคำสั่ง เช่น เวลาที่ใช้ในการรับส่งข้อมูลด้วยคำสั่งของเอ็มพีไอ หรือเวลาที่ใช้กระจายภาระงานด้วยคำสั่งของโอเพนเอ็มพี (2) แผนภาพการเรียกใช้ฟังก์ชันของโปรแกรม และ (3) สมรรถนะของแต่ละส่วนบนระบบคอมพิวเตอร์ เช่น เทรด โพรเซส และโหนด

งานวิจัยของวูล์ฟและมอห์ร (Wolf & Mohr, 2003) ได้นำเสนอการใช้โคแจกวิเคราะห์สมรรถนะของโปรแกรมเชิงขนานแบบผสมผสานระหว่างเอ็มพีไอและโอเพนเอ็มพี สองโปรแกรม

---

[6] The Kit for Objective Judgement And Knowledge-based detection of performance bottlenecks

ประกอบด้วย โปรแกรมคำนวณพยากรณ์อากาศรีโม (REMO) และโปรแกรมทางการแพทย์ซวีปทรีดี (SWEEP3D) โดยนำไปประมวลผลบนเครื่องอินเทลซีนอนคลัสเตอร์จำนวน 32 หน่วยประมวลผล ผลการวิเคราะห์พบว่า โปรแกรมรีโมมีเวลาเดินเครื่องเปล่าบนคลัสเตอร์เกิดขึ้นถึงร้อยละ 51.6 เนื่องจากพบปัญหาคอขวดบนส่วนประมวลผลแบบลำดับของพื้นที่โปรแกรม ที่เขียนด้วยเอพีไอของโอเพนเอ็มพี ซึ่งพื้นที่ในส่วนดังกล่าวจะมีเทรดหลักประมวลผลเพียงเทรดเดียว จึงทำให้เทรดย่อยที่อยู่หน่วยประมวลผลอื่นหยุดเพื่อรอประมวลผล ส่วนโปรแกรมซวีปทรีดีนั้น พบว่า การเขียนโปรแกรมแบบผสมผสานทำให้เกิดปัญหาการทำงานร่วมกัน ระหว่างโพรเซสของเอ็มพีไอ กับเทรดของโอเพนเอ็มพี ส่งผลให้เกิดเวลาเดินเครื่องเปล่าขึ้นบนโปรแกรมถึงร้อยละ 37 จากงานวิจัยของวูล์ฟนี้แสดงให้เห็นถึง การนำเสนอข้อมูลในลักษณะของแผนภาพการเรียกใช้ฟังก์ชั่น ทำให้ผู้ใช้ทราบถึงส่วนของโปรแกรมที่ทำให้เกิดปัญหาคอขวด ซึ่งปัญหาดังกล่าวนี้ทำให้บางหน่วยประมวลผลต้องเดินเครื่องเปล่า ส่งผลให้โปรแกรมสูญเสียสมรรถนะการประมวลผล

3.3.2.2 ทาว (TAU ย่อมาจาก Tuning and Analysis Utilities)

ทาว (Shende & Malony, 2006) ถูกพัฒนาโดยนักวิจัยจากมหาวิทยาลัยโอเรกอน เพื่อใช้เป็นเครื่องมือวิเคราะห์สมรรถนะโปรแกรมเชิงขนานบนระบบคอมพิวเตอร์ที่มีขนาดใหญ่ โดยสามารถวิเคราะห์โปรแกรมเชิงขนานได้หลายรูปแบบ เช่น เอ็มพีไอ โอเพนเอ็มพี เอสเอชเมม (SHMEM) และชาร์ม (Charm) และสามารถรองรับภาษาโปรแกรมได้หลายภาษา ได้แก่ ภาษาซี/ซีพลัสพลัส จาวา ไพธอน และฟอร์แทรน รวมถึงการเขียนโปรแกรมเชิงวัตถุของภาษาเหล่านั้นได้ ทาวสามารถปรับโหมดการทำงานได้ทั้งแบบโพรไฟลิ่งและเทรซซิง และสามารถนำเสนอข้อมูลได้หลายรูปแบบ เช่น ไฟล์ข้อความ กราฟ ไดอะแกรมเชิงเวลา และแผนภาพการเรียกฟังก์ชัน

มอร์ริสและคณะ (Morris *et al.*, 2007) ได้พัฒนาไลบรารีของทาวเพิ่มเติม เพื่อใช้วิเคราะห์การประมวลผลแบบขนานที่มีลูปซ้อนกันมากกว่าหนึ่งระดับ จากการวิเคราะห์โปรแกรมคำนวณอากาศพลศาสตร์ทีเอฟเอส เพื่อเปรียบเทียบการประมวลผลแบบขนานที่มีลูปซ้อนและไม่ซ้อนกัน ซึ่งผลจากการวิเคราะห์พบว่า เมื่อประมวลผลบนเครื่องซันไฟร์ 25 เค (Sun Fire 25 K) โปรแกรมทีเอฟเอสที่ประมวลผลในลูปซ้อนกันมากกว่าหนึ่งระดับ สามารถให้ค่าสปีดอัพสูงถึง 21 เท่า เมื่อเทียบกับโปรแกรมที่ประมวลผลแบบขนานที่ระดับลูปเพียงหนึ่งระดับ จากผลการวิเคราะห์ชี้ให้เห็นว่า โปรแกรมแบบโอเพนเอ็มพีมีโอกาสที่จะเพิ่มสมรรถนะการประมวลผลได้ โดยใช้เทคนิคการประมวลผลแบบขนานที่มีลูปซ้อนกันมากกว่าหนึ่งระดับ

จากงานวิจัยที่กล่าวถึงในหัวข้อนี้ จะเห็นได้ว่าเครื่องมือวิเคราะห์สมรรถนะที่มีในปัจจุบันสามารถนำข้อมูลเชิงสมรรถนะได้หลายรูปแบบ เช่น เวลาประมวลผล เวลาส่วนเกิน เวลาเดินเครื่องเปล่า ข้อมูลเรียกใช้ฟังก์ชันภายในโปรแกรม และความสมดุลของการกระจายภาระงาน ซึ่งข้อมูลเหล่านี้โปรแกรมเมอร์สามารถนำไปใช้เพื่อปรับปรุงสมรรถนะของโปรแกรมได้ อย่างไรก็ตาม ดังที่ได้กล่าวไว้ในหัวข้อก่อนหน้านี้ การใช้ประโยชน์พื้นที่แคชเป็นปัจจัยสำคัญที่ส่งผลต่อสมรรถนะของโปรแกรม แต่ปรากฏว่าเครื่องมือที่ได้กล่าวถึงในหัวข้อนี้ ยังนำเสนอข้อมูลการใช้ประโยชน์พื้นที่แคชจำกัด เนื่องจากการเก็บข้อมูลการทำงานของแคชนั้นต้องเก็บข้อมูลที่ระดับฮาร์ดแวร์ และใช้เวลาการเก็บข้อมูลที่นาน ขณะที่เครื่องมือเหล่านี้ต้องการความเร็วในการนำเสนอแก่ผู้ใช้ มีงานวิจัยหลายชิ้นที่วิเคราะห์การใช้ประโยชน์พื้นที่แคชบนระบบจำลอง เนื่องจากระบบจำลองสามารถให้ผู้ใช้ปรับเปลี่ยนสถาปัตยกรรมของแคชภายในตัวเองได้ จึงทำให้สามารถวิเคราะห์การทำงานของแคชได้หลากหลายโดยที่ไม่ต้องเปลี่ยนระบบคอมพิวเตอร์ ซึ่งในหัวข้อถัดไปจะกล่าวถึง ระบบจำลองที่นำมาใช้วิเคราะห์การใช้ประโยชน์พื้นที่แคช

3.4 การเพิ่มสมรรถนะโปรแกรมแบบโอเพนเอ็มพีด้วยระบบจำลอง

จากที่กล่าวไว้ในบทที่ 2 เทคนิคการวิเคราะห์สมรรถนะด้วยระบบจำลองมีข้อได้เปรียบ ที่สามารถใช้วิเคราะห์ระบบคอมพิวเตอร์ต้นแบบหรือที่อยู่ในช่วงกำลังพัฒนาได้ การวิเคราะห์ด้วยระบบจำลองจึงมักถูกนำไปใช้ในงานออกแบบหรือการวิเคราะห์สถาปัตยกรรมระบบคอมพิวเตอร์รวมถึงสถาปัตยกรรมของแคช มีงานวิจัยหลายชิ้นได้นำเสนอระบบจำลองเพื่อใช้วิเคราะห์การใช้ประโยชน์พื้นที่แคช บนโปรแกรมแบบโอเพนเอ็มพี ระบบจำลองเหล่านี้สามารถให้ผู้ใช้ปรับแต่งสถาปัตยกรรมของแคช เช่น ขนาดของแคช และขนาดของแคชบล็อกได้ รวมไปถึงสามารถปรับเปลี่ยนวิธีการทำงานบนแคช เช่น การปรับข้อมูล การยกเลิกข้อมูล และกลไกรักษาความสอดคล้องของข้อมูลในแคชได้ เพื่อให้ผู้ใช้สามารถเปรียบเทียบการใช้ประโยชน์พื้นที่แคชบนแต่ละสถาปัตยกรรมได้ง่ายขึ้น ซึ่งจะได้กล่าวถึงงานวิจัยสามชิ้น ที่นำเสนอระบบจำลองเพื่อใช้วิเคราะห์การใช้ประโยชน์พื้นที่แคช บนโปรแกรมแบบโอเพนเอ็มพี

มะเริงสิทธิ์และอิบเบตต์ (Marurngsith & Ibbett, 2004) ได้นำเสนอ ดีซิมคลัสเตอร์ (DSiMCluster) ซึ่งเป็นระบบจำลองที่ใช้วิเคราะห์ประสิทธิภาพการทำงานของหน่วยความจำบนเครื่องคลัสเตอร์แบบใช้หน่วยความจำกระจายร่วมกัน การประมวลผลโปรแกรมแบบใช้หน่วยความจำร่วม (โปรแกรมแบบโอเพนเอ็มพี) บนเครื่องคลัสเตอร์ที่มีสถาปัตยกรรมดังกล่าวนั้น

มีข้อจำกัดของการเข้าถึงข้อมูลข้ามโหนดที่ต้องใช้เวลานาน ทำให้โปรแกรมสูญเสียสมรรถนะการประมวลผล ดีซิมคลัสเตอร์จึงถูกพัฒนาเพื่อใช้วิเคราะห์คุณลักษณะการเข้าถึงข้อมูลบนหน่วยความจำ รวมถึงสัดส่วนการเข้าถึงข้อมูลข้ามโหนดของโปรแกรม โดยผู้ใช้สามารถปรับแต่งค่าพารามิเตอร์ทางสถาปัตยกรรมเครื่องคลัสเตอร์ได้หลายส่วน เช่น จำนวนโหนด จำนวนหน่วยประมวลผลต่อโหนด สถาปัตยกรรมของแคช และกลไกรักษาความสอดคล้องของแคช

มะเริงสิทธิ์ได้นำเสนอการใช้ระบบจำลองดังกล่าวนี้ วิเคราะห์โปรแกรมย่อยแอลยูของชุดโปรแกรมเอ็นพีบี บนเครื่องคลัสเตอร์จำนวน 4 โหนด เพื่อทดลองเปรียบเทียบสมรรถนะของโปรแกรมดังกล่าว เมื่อต้องจัดรูปแบบอาร์เรย์ข้อมูลเป็นแบบลำดับแถวเป็นหลักและลำดับคอลัมน์เป็นหลัก ผลการวิเคราะห์พบว่า โปรแกรมแอลยูที่จัดรูปแบบอาร์เรย์แบบคอลัมน์เป็นหลักมีอัตราการไม่พบข้อมูลน้อยกว่าร้อยละ 65.5 มีการใช้ประโยชน์พื้นที่แคชสูงกว่า 47.4 และมีสมรรถนะการประมวลผลสูงกว่า เมื่อเทียบกับโปรแกรมที่จัดรูปแบบอาร์เรย์แบบแถวเป็นหลัก เนื่องจากการจัดรูปแบบลำดับแถวเป็นหลักทำให้เกิดการไม่พบข้อมูลร่วมใช้ไม่จริงบนแคชที่สูงกว่า ทำให้ต้องแลกเปลี่ยนข้อมูลระหว่างโหนดเพิ่มสูงขึ้น ซึ่งต้องใช้เวลาการเข้าถึงข้อมูลเป็นเวลานาน งานวิจัยดังกล่าวนี้แสดงให้เห็นว่า การจัดรูปแบบของข้อมูลให้สอดคล้องกับสถาปัตยกรรมของแคช มีผลกระทบต่อสมรรถนะการประมวลผลของโปรแกรม ซึ่งระบบจำลองเป็นเครื่องมือชนิดหนึ่งที่ช่วยนำเสนอข้อมูลที่มีผลกระทบดังกล่าวนั้นได้

จากปัญหาการเข้าถึงข้อมูลบนหน่วยความจำข้ามโหนด บนเครื่องคลัสเตอร์แบบใช้หน่วยความจำกระจายร่วมกันนั้น เทาและคณะ (Tao *et al*., 2005) ได้พัฒนาและนำเสนอ ซิมที/โอเอ็มพี (SIMT/OMP) เพื่อใช้วิเคราะห์ปัญหาดังกล่าวบนโปรแกรมแบบโอเพนเอ็มพีโดยเฉพาะ ซิมที/โอเอ็มพีสามารถให้ผู้ใช้ปรับแต่งค่าพารามิเตอร์ได้หลายส่วนในลักษณะเช่นเดียวกับดีซิมคลัสเตอร์ นอกจากนี้ ซิมที/โอเอ็มพียังมีเครื่องมือช่วยให้โปรแกรมเมอร์ใช้ปรับวิธีการแมปข้อมูลที่ใช้ในโปรแกรมให้เหมาะสมกับสถาปัตยกรรมของแคช ซึ่งจะส่งผลให้อัตราการพบข้อมูลในแคชระดับสุดท้ายของแต่ละโหนดเพิ่มสูงขึ้น ทำให้อัตราการเข้าถึงข้อมูลข้ามโหนดลดลง

เทาได้นำเสนอการใช้ซิมที/โอเอ็มพี เพื่อวิเคราะห์อัตราการไม่พบข้อมูล และสัดส่วนการเข้าถึงข้อมูลข้ามโหนด ของชุดโปรแกรมเอ็นพีบีและชุดโปรแกรมสแปลช-2 (Splash-2) พบว่าบนเครื่องคลัสเตอร์ที่มีหน่วยประมวลผล 32 หน่วยนั้น โปรแกรมทั้งหมดมีสัดส่วนการเข้าถึงข้อมูลข้ามโหนดสูงถึงร้อยละ 90 แต่เมื่อใช้เครื่องมือช่วยปรับการแมปข้อมูลของซิมที/โอเอ็มพี ส่งผลให้โปรแกรมมีอัตราการไม่พบข้อมูลบนแคชระดับที่ 2 ลดลง และมีสัดส่วนการเข้าถึงข้อมูลข้ามโหนดลดลงถึงร้อยละ 63 ซึ่งผู้วิจัยได้แสดงความเห็นว่า การลดสัดส่วนการเข้าถึงข้อมูลข้ามโหนดเป็น

ปัจจัยสำคัญที่ช่วยเพิ่มสมรรถนะการประมวลผลบนเครื่องคลัสเตอร์ขนาดใหญ่ นอกจากนี้ วิธีการแมปข้อมูลที่เหมาะสมกับสถาปัตยกรรมของแคช นอกจากจะช่วยลดสัดส่วนการเข้าถึงข้อมูลข้ามโหนดแล้ว ยังช่วยลดการทำงานของกลไกรักษาความสอดคล้องของข้อมูล บนแคชของแต่ละหน่วยประมวลผลได้อีกด้วย

ดังที่กล่าวไว้ในบทที่ 2 กลไกรักษาความสอดคล้องของข้อมูลในแคช เป็นปัจจัยสำคัญที่มีผลต่อสมรรถนะการประมวลผลของระบบคอมพิวเตอร์เชิงขนาน มาราธีและมูลเลอร์ (Marathe & Mueller, 2007) จึงนำเสนอ ซีซีซิม (ccSIM ย่อมาจาก Cache-coherence simulation) ซึ่งเป็นระบบจำลองที่ใช้ศึกษาผลของกลไกรักษาความสอดคล้องของข้อมูลในแคช ที่มีต่อสมรรถนะการประมวลผลของโปรแกรมแบบโอเพนเอ็มพี โดยซีซีซิมจะจำแนกการไม่พบข้อมูลเนื่องจากการใช้ข้อมูลร่วมกัน ทั้งชนิดร่วมใช้จริงและไม่จริงออกจากการไม่พบข้อมูลชนิดอื่น เพื่อแสดงให้เห็นถึงผลกระทบการทำงานของกลไกรักษาความสอดคล้องของข้อมูลที่มีต่อโปรแกรมแบบโอเพนเอ็มพี

มาราธีได้นำเสนอการใช้ซีซีซิม เพื่อวิเคราะห์อัตราการไม่พบข้อมูลเนื่องจากการใช้ข้อมูลร่วมกันของโปรแกรมทางวิทยาศาสตร์ที่ใช้งานจริงสามโปรแกรม[7] ประกอบด้วย เอ็นบีเอฟ ไอ อาร์เอส และเอสเอ็มจี2000 โดยจำลองการทำงานของหน่วยประมวลผลพาวเวอร์ 3 ของไอบีเอ็ม ที่ใช้กลไกรักษาความสอดคล้องของข้อมูลแบบเมซี (MESI) ผลการวิเคราะห์ พบว่า บนโปรแกรมเอ็นบีเอฟมีอัตราไม่พบข้อมูลร่วมใช้จริงบนแคชระดับที่ 2 สูงถึงร้อยละ 80 ของการไม่พบข้อมูลทั้งหมด ส่วนบนโปรแกรมไอาร์เอสและเอสเอ็มจี2000 พบว่า มีอัตราการไม่พบข้อมูลร่วมใช้ไม่จริงที่แคชระดับที่ 2 ร้อยละ 40 และ 90 ตามลำดับ จากงานวิจัยของมาราธีแสดงให้เห็นว่าพบว่ากลไกรักษาความสอดคล้องของข้อมูลในแคชนั้น จะส่งผลต่อสมรรถนะของโปรแกรม โดยจะทำให้เกิดการไม่พบข้อมูลเนื่องจากการใช้ร่วมกันทั้งสองชนิด ซึ่งโปรแกรมเมอร์สามารถนำข้อมูลดังกล่าวไปปรับปรุงสมรรถนะของโปรแกรมให้สูงขึ้นได้

จากงานวิจัยที่กล่าวถึงในหัวข้อนี้ จะเห็นได้ว่า ระบบจำลองเป็นอีกทางเลือกหนึ่งที่สามารถนำมาใช้เป็นเครื่องมือวิเคราะห์สมรรถนะของโปรแกรม ที่ได้รับผลกระทบจากการเข้าถึงข้อมูลบนทั้งหน่วยความจำหลักและแคช จากตัวอย่างงานวิจัยที่กล่าวถึง ได้สาธิตแนวทางการนำข้อมูลการใช้ประโยชน์พื้นที่แคชไปปรับปรุงสมรรถนะของโปรแกรม ซึ่งจะเห็นได้ว่าระบบจำลองสามารถวิเคราะห์และนำเสนอข้อมูลการใช้ประโยชน์พื้นที่แคช ที่มีความชัดเจนและหลากหลาย

---

[7] NBF ย่อมาจาก Non-bond Force เป็นโปรแกรมการจำลองโมเลกุล, IRS ย่อมาจาก Implicit Radiation Solver และ SMG2000 ย่อมาจาก Semicoarsening Grid Solver ทั้งสองโปรแกรมหลังเป็นส่วนหนึ่งของชุดโปรแกรมของแอสกี (ASCI)

กว่าเครื่องมือวิเคราะห์สมรรถนะบนระบบจริงที่ได้กล่าวถึงในหัวข้อก่อนหน้านี้ และระบบจำลองยังสามารถให้ผู้ใช้ปรับแต่งสถาปัตยกรรมของแคชได้ง่าย ดังนั้น ระบบจำลองจึงเป็นทางเลือกที่เหมาะสมมากกว่า เพื่อใช้เป็นเครื่องมือวิเคราะห์และนำเสนอข้อมูลการใช้ประโยชน์พื้นที่แคชสำหรับโปรแกรมแบบโอเพนเอ็มพี

3.5 แนวคิดใหม่ของพี-สปาซึ่งต่อยอดจากงานวิจัยที่เกี่ยวข้อง

จากการสำรวจงานวิจัยที่เกี่ยวข้องกับโปรแกรมแบบโอเพนเอ็มพี ตั้งแต่ปี พ.ศ.2540 งานวิจัยในหัวข้อที่ 3.1 แสดงให้เห็นว่า การประมวลผลแบบขนานที่ระดับลูปเป็นส่วนพื้นที่ที่สำคัญของโปรแกรม วิธีการแจกตัวนับลูปเพื่อประมวลผลแบบขนาน ด้วยคำสั่งกำกับคอมไพเลอร์ของโอเพนเอ็มพีนั้น มีผลกระทบต่อสมรรถนะการประมวลผลของโปรแกรม ซึ่งสามารถอธิบายผลกระทบดังกล่าวได้ด้วย เวลาประมวลผล เวลาส่วนเกิน และการใช้ประโยชน์พื้นที่แคช ทั้งนี้วิธีการแจกตัวนับลูปแต่ละวิธีมีความเหมาะสมกับคุณลักษณะของโปรแกรม และสถาปัตยกรรมคอมพิวเตอร์แตกต่างกัน อย่างไรก็ตาม โปรแกรมเมอร์ไม่สามารถปรับความสมดุลการแจกตัวนับลูปได้อย่างเต็มที่ เนื่องจากโปรแกรมแบบโอเพนเอ็มพีใช้คอมไพเลอร์สร้างโค้ดเพื่อประมวลผลแบบขนานโดยอัตโนมัติ แม้ว่าจากงานวิจัยในหัวข้อที่ 3.2 จะแสดงให้เห็นว่า หากเลือกวิธีการแจกตัวนับลูปที่ทำให้เกิดการใช้ประโยชน์พื้นที่แคชอย่างเต็มประสิทธิภาพ ก็จะช่วยทำให้สมรรถนะการประมวลผลของโปรแกรมสูงขึ้นด้วย ซึ่งโปรแกรมเมอร์สามารถใช้ข้อมูลการใช้ประโยชน์พื้นที่แคช เพื่อตัดสินใจเลือกวิธีการแจกตัวนับลูปที่จะทำให้โปรแกรมมีสมรรถนะสูงที่สุดได้ ดังนั้น เครื่องมือที่สามารถนำเสนอข้อมูลการใช้ประโยชน์พื้นที่แคชที่ได้รับผลจากวิธีการแจกตัวนับลูป จึงมีความจำเป็นสำหรับการเขียนโปรแกรมแบบโอเพนเอ็มพี เพื่อให้ได้สมรรถนะตามที่ต้องการ

จากการสำรวจงานวิจัยที่นำเสนอเครื่องมือวิเคราะห์สมรรถนะสำหรับโปรแกรมเชิงขนาน รวมถึงโปรแกรมแบบโอเพนเอ็มพี บนระบบจริง ที่ได้กล่าวถึงในหัวข้อที่ 3.3 พบว่า เครื่องมือเหล่านี้สามารถรองรับภาษาโปรแกรม ระบบปฏิบัติการ และสถาปัตยกรรมคอมพิวเตอร์ได้หลากหลาย พร้อมทั้งมีรูปแบบการนำเสนอที่มีประสิทธิภาพและง่ายต่อการทำความเข้าใจของผู้ใช้ เช่น ไฟล์ข้อความ กราฟ ไดอะแกรมเชิงเวลา และแผนภาพการเรียกใช้ฟังก์ชัน ดังที่สรุปไว้ในตารางที่ 3.1 อย่างไรก็ตาม เครื่องมือที่ได้กล่าวถึงเหล่านั้นยังนำเสนอข้อมูลการใช้ประโยชน์พื้นที่แคชได้จำกัด เนื่องจากการพัฒนาเครื่องมือเหล่านั้นต้องการนำเสนอข้อมูลให้แก่ผู้ใช้ได้อย่างรวดเร็ว ขณะที่การวิเคราะห์การใช้ประโยชน์พื้นที่แคชนั้นต้องเก็บข้อมูลระดับฮาร์ดแวร์ซึ่งต้องใช้เวลานาน

แม้ว่ามีงานวิจัยที่นำเสนอระบบจำลองเพื่อใช้วิเคราะห์การใช้ประโยชน์พื้นที่แคช ดังที่กล่าวถึงในหัวข้อที่ 3.4 อย่างไรก็ตาม จากตารางที่ 3.1 แสดงให้เห็นว่า ระบบจำลองเหล่านั้นยังคงนำเสนอการใช้ประโยชน์พื้นที่แคชได้ไม่ครบถ้วน บางเครื่องมือ (ดีซิมคลัสเตอร์) ต้องแปลงโค้ดของโปรแกรมให้อยู่ในรูปแบบที่ระบบจำลองกำหนดไว้ ไม่สามารถประมวลผลโปรแกรมที่ต้องการวิเคราะห์ได้โดยตรง และบางเครื่องมือ (ซิมที/โอเอ็มพี) ต้องทำงานภายใต้คอมไพเลอร์ที่กำหนดไว้เท่านั้น นอกจากนี้ ระบบจำลองเหล่านั้นจะผูกติดอยู่กับแพลตฟอร์มไม่สามารถทำงานย้ายข้ามแพลตฟอร์มได้ ทำให้ประสบปัญหาเมื่อต้องการเปรียบเทียบสมรรถนะข้ามแพลตฟอร์ม

จากความสำคัญของการประมวลผลแบบขนานที่ระดับลูปของโปรแกรมแบบโอเพนเอ็มพี เนื่องจากเป็นส่วนที่ใช้เวลาประมวลผลสูงที่สุด และยังส่งผลต่อการใช้ประโยชน์พื้นที่แคช ซึ่งปัจจัยสำคัญต่อสมรรถนะการประมวลผลของโปรแกรม รวมถึงปัจจุบันยังขาดเครื่องมือที่ใช้วิเคราะห์การใช้ประโยชน์พื้นที่แคช ที่ได้รับผลกระทบจากการประมวลผลแบบขนานที่ระดับลูปโดยตรง ทำให้โปรแกรมเมอร์ทั่วไปยังขาดข้อมูล เพื่อการตัดสินใจเลือกใช้คำสั่งกำกับการแจกตัวนับลูปที่เหมาะสมกับคุณลักษณะของโปรแกรมและสถาปัตยกรรมคอมพิวเตอร์ ดังนั้นโครงงานวิจัยนี้จึงมีวัตถุประสงค์พัฒนาเครื่องมือวิเคราะห์สมรรถนะที่ชื่อว่าพี-สปา เพื่อใช้เป็นเครื่องมือวิเคราะห์ที่สามารถวิเคราะห์การใช้ประโยชน์พื้นที่แคช ที่ได้รับผลกระทบอันเนื่องมาจากวิธีการแจกตัวนับลูปของคำสั่งกำกับคอมไพเลอร์ โดยกำหนดเป้าหมายการพัฒนาให้เครื่องมือดังกล่าวนี้สามารถนำเสนอข้อมูลอัตราการไม่พบข้อมูลบนแคช ชนิดของการไม่พบข้อมูลในแคช คุณสมบัติพื้นที่เก็บข้อมูล และเวลาประมวลผลของโปรแกรม ซึ่งพี-สปาจะเป็นอีกทางเลือกที่ช่วยให้โปรแกรมเมอร์สามารถพัฒนาโปรแกรมแบบโอเพนเอ็มพี ให้มีสมรรถนะการประมวลผลที่สูงขึ้น