บทที่ 3

วิธีการดำเนินงานวิจัย

ในบทนี้ จะนำเสนอรายละเอียดวิธีการดำเนินงานวิจัยของงานวิจัยนี้ โดยส่วนแรกของบทจะกล่าวถึงลำดับขั้นตอนการดำเนินงานวิจัย ซึ่งกล่าวถึงภาพรวมของการศึกษาและการทดลองทั้งหมดในงานวิจัย ส่วนถัดมาของบทจะอธิบายในรายละเอียดของการดำเนินงานวิจัย โดยจะประกอบด้วย การทดสอบเวลาการทำงานของฟังก์ชันในโปรแกรม MapServer การตรวจสอบขั้นตอนการทำงานของโปรแกรม MapServer โครงสร้างการทำงานของระบบ แนวทางการออกแบบ อัลกอริทึมแบบขนาน และ การวัดและวิเคราะห์ผลจากการทดลอง

3.1 ขั้นตอนการดำเนินงานวิจัย

ลำดับขั้นตอนการดำเนินการวิจัยที่กระทำในงานวิจัยนี้ทั้งหมด สามารถจำแนกเป็นรายการได้ดังต่อไปนี้

3.1.1 ศึกษาและทดสอบเวลาการทำงานของฟังก์ชันในระบบโปรแกรม MapServer

ขั้นตอนแรกของการทำงานวิจัยนี้ จะเริ่มต้นที่การทดสอบเวลาการทำงานของระบบโปรแกรม MapServer ในแต่ละฟังก์ชัน โดยจุดมุ่งหมายหลักของการทดสอบเวลาการทำงานนี้ ก็เพื่อที่จะหาฟังก์ชันของ MapServer ที่เหมาะสมสำหรับใช้ในงานวิจัยมากที่สุด ซึ่งฟังก์ชันที่เหมาะสมสำหรับงานวิจัย จะมีลักษณะคือ เป็นฟังก์ชันที่มีการคำนวณและประมวลผลข้อมูลสูงและเวลาการตอบสนองของฟังก์ชันต่อผู้ใช้งานค่อนข้างช้า เหตุที่เลือกฟังก์ชันลักษณะนี้มาใช้ในงานวิจัย เนื่องจากฟังก์ชันลักษณะนี้ส่วนใหญ่ในระบบ GIS จะเป็นฟังก์ชันสำหรับการวิเคราะห์ข้อมูลเชิงพื้นที่ ซึ่งต้องใช้เวลาการประมวลผลค่อนข้างมาก และเมื่อข้อมูลมีขนาดเพิ่มขึ้น เวลาการประมวลผลก็จะเพิ่มขึ้นตามไปด้วย ดังนั้นเมื่อเลือกฟังก์ชันเหล่านี้มาประยุกต์ใช้เทคนิคการประมวลผลแบบขนานเพิ่มประสิทธิภาพการทำงาน จะทำให้เห็นผลความแตกต่างระหว่างฟังก์ชันแบบเดิมและฟังก์ชันหลังการประยุกต์ใช้ตามงานวิจัยค่อนข้างชัดเจน รวมทั้งยังเป็นประโยชน์อย่าง

มากสำหรับการพัฒนาประสิทธิภาพการทำงานของฟังก์ชันการวิเคราะห์ข้อมูลให้สามารถใช้งานบนระบบ Web-based GIS ได้เป็นอย่างดี

3.1.2 ตรวจสอบขั้นตอนการทำงานของระบบโปรแกรม MapServer

หลังจากที่ทำการทดสอบเวลาการทำงานของแต่ละฟังก์ชัน และได้ฟังก์ชันที่เหมาะสมสำหรับใช้ในงานวิจัยเรียบร้อยแล้ว ขั้นตอนต่อไปก็คือ การตรวจสอบโปรแกรม MapServer ว่ามีลำดับขั้นตอนรวมไปถึงวิธีการทำงานของโปรแกรมเป็นอย่างไรบ้าง โดยจะเน้นไปที่ฟังก์ชันที่เลือกมาใช้ในงานวิจัยข้างต้นเป็นหลัก สำหรับวิธีการตรวจสอบขั้นตอนการทำงานของโปรแกรม MapServer นั้น ในงานวิจัยนี้ใช้เครื่องมือ gprof ร่วมกับเครื่องมือ GDB ซึ่งเครื่องมือ gprof จะใช้สำหรับการตรวจสอบการทำงานของโปรแกรมรวมทั้งเวลาที่แต่ละฟังก์ชันของ MapServer ใช้ ส่วนเครื่องมือ GDB จะเป็นเครื่องมือที่ใช้ในการตรวจสอบลำดับการทำงาน และการเรียกฟังก์ชันภายในของโปรแกรม MapServer

3.1.3 ศึกษาความเป็นไปได้ในการนำเอาเทคนิคการพัฒนาโปรแกรมแบบขนานมาประยุกต์ใช้

ขั้นตอนการดำเนินงานวิจัยในส่วนนี้ เป็นส่วนที่ต่อเนื่องมาจากสองขั้นตอนข้างต้น หลังจากที่ได้ฟังก์ชันที่เหมาะสมสำหรับงานวิจัย พร้อมทั้งศึกษารูปแบบและขั้นตอนการทำงานของฟังก์ชันดังกล่าวแล้ว จะเข้าสู่ขั้นตอนของการวิเคราะห์ว่า รูปแบบการทำงานของโปรแกรมฟังก์ชันดังกล่าวสามารถประยุกต์นำเอาเทคนิคการพัฒนาโปรแกรมแบบขนานมาใช้แก้ปัญหาได้หรือไม่ ซึ่งวิเคราะห์หรือการศึกษาความเป็นไปได้นี้ จะเน้นไปที่การศึกษาถึงความสัมพันธ์ของการใช้งานข้อมูล (Data Dependency Analysis) ของโปรแกรม ในประเด็นที่ว่า ตัวแปรต่างๆ ที่กำหนดภายในโปรแกรมมีความสัมพันธ์กันอย่างไร สามารถแก้ไขความสัมพันธ์ดังกล่าวเพื่อทำการประมวลผลแบบขนานได้หรือไม่ นอกจากนั้นยังทำการศึกษาและวิเคราะห์ความสัมพันธ์ของลำดับการเรียกใช้ข้อมูลก่อนหลังในแต่ละคำสั่งภายในโปรแกรมอีกด้วย

3.1.4 ออกแบบโครงสร้างและรูปแบบการทำงานของระบบโปรแกรม MapServer แบบใหม่

เมื่อทำการวิเคราะห์ความเป็นไปได้และสรุปได้ว่าสามารถปรับปรุงโปรแกรม MapServer ให้ใช้เทคนิคการประมวลผลแบบขนานได้แล้ว ขั้นตอนต่อไปคือการออกแบบ

โครงสร้างและรูปแบบการทำงานให้กับระบบโปรแกรม MapServer ที่จะพัฒนาปรับปรุงขึ้นใหม่ ซึ่งสำหรับขั้นตอนนี้การออกแบบสามารถจำแนกเป็นการออกแบบในส่วนต่างๆ ได้แก่ การออกแบบโครงสร้างสถาปัตยกรรมของระบบ การเลือกใช้เครื่องมือและซอฟต์แวร์ต่างๆ ที่ใช้ในการทำงานของระบบ การออกแบบวิธีการทำงานรวมทั้งวิธีการเข้าถึงข้อมูลของระบบ การออกแบบโครงสร้างของโปรแกรมและส่วนของโปรแกรม (Module) ที่จำเป็นต้องพัฒนา และ การออกแบบแนวทางการกระจายงานและการประมวลผลแบบขนาน โดยผลลัพธ์ที่ได้ในขั้นตอนการทำงานวิจัยนี้ จะใช้สำหรับเป็นแนวทางในการพัฒนาระบบโปรแกรม MapServer แบบใหม่ในงานวิจัยต่อไป

3.1.5 พัฒนาระบบโปรแกรม MapServer โดยประยุกต์ใช้เทคนิคการประมวลผลแบบขนาน

ขั้นตอนการดำเนินงานวิจัยในส่วนนี้ เป็นขั้นตอนของการลงมือพัฒนาระบบโปรแกรม MapServer แบบใหม่ เพื่อพิสูจน์หรือยืนยันแนวคิดของงานวิจัย ซึ่งก็คือการใช้เทคนิคการประมวลผลแบบขนานมาปรับปรุงประสิทธิภาพของระบบ Web-Based GIS ให้ดียิ่งขึ้น ซึ่งขั้นตอนนี้จะเกิดขึ้นหลังจากที่ได้ทำการออกแบบโครงสร้างและรูปแบบการทำงานของระบบโปรแกรม MapServer แบบใหม่ขึ้นมาเรียบร้อยแล้ว โดยในการทำงานของขั้นตอนนี้จะนำเอาโครงสร้างระบบที่ออกแบบใหม่ มาพัฒนาปรับปรุงโปรแกรม MapServer แบบดั้งเดิมที่พัฒนาโดยใช้เทคนิคการเขียนโปรแกรมแบบลำดับ (Sequential) ให้สามารถทำงานและประมวลผลในรูปแบบของการประมวลผลแบบขนานได้

3.1.6 ทดสอบประสิทธิภาพระบบโปรแกรม MapServer ที่พัฒนาขึ้นใหม่ เทียบกับระบบโปรแกรม MapServer แบบดั้งเดิม

หลังจากที่พัฒนาระบบโปรแกรม MapServer โดยประยุกต์ใช้เทคนิคการประมวลผลแบบขนานสำเร็จเรียบร้อย ขั้นตอนการทำงานวิจัยต่อไปคือ นำระบบโปรแกรมที่พัฒนาขึ้นใหม่มาทดสอบการทำงานเปรียบเทียบประสิทธิภาพ ความเร็วในการประมวลผลกับระบบโปรแกรมเดิม โดยจะนำทั้งระบบโปรแกรม MapServer ที่พัฒนาขึ้นใหม่ และระบบโปรแกรม MapServer แบบเดิม มาทำงานในสถานการณ์และสภาพแวดล้อมเดียวกัน ทดสอบการประมวลผลและประสิทธิภาพที่เพิ่มขึ้นจากการประยุกต์ใช้เทคนิคการประมวลผลแบบขนาน โดยการเพิ่มจำนวนหน่วยประมวลผลเป็นทวีคูณ จาก 1 หน่วยประมวลผล เป็น 2 หน่วยประมวลผล และทวีคูณเป็น 4 หน่วยประมวลผลเพิ่มไปเรื่อยๆ พร้อมทั้งวัดผลการทำงานจากเวลาการตอบสนองของระบบ

โปรแกรม (Response Time) และเวลาการประมวลผลทั้งหมดของระบบโปรแกรม (Elapse Wall-clock times) ที่ได้ในการประมวลผลแต่ละครั้ง

3.1.7 วิเคราะห์ผลการทดลอง

การวิเคราะห์ผลการทดลอง เป็นขั้นตอนสุดท้ายของการดำเนินการวิจัย ซึ่งในขั้นตอนนี้จะนำเอาผลลัพธ์ของเวลาการตอบสนองของระบบและเวลาการประมวลผลทั้งหมดที่ได้จากขั้นตอนการทดสอบประสิทธิภาพของระบบ มาทำการวิเคราะห์ในประเด็นที่ว่า ระบบโปรแกรมใหม่มีเวลาการตอบสนองมากน้อยเท่าใด เมื่อเทียบกับระบบเดิม (Speedup) และถ้าคำนึงถึงประสิทธิภาพของระบบ ระบบใหม่มีประสิทธิภาพในการใช้งานหน่วยประมวลผลที่เพิ่มมากขึ้นเป็นอย่างไร (Processor Efficiency) ซึ่งประเด็นที่ใช้สำหรับวิเคราะห์ดังกล่าวคือ ตัวชี้วัด (Indicator) ที่จะแสดงถึงผลสำเร็จของการเพิ่มประสิทธิภาพของระบบโปรแกรมที่เสนอในงานวิจัย โดยการวิเคราะห์ในงานวิจัยนี้ จะแสดงผลลัพธ์ของการวิเคราะห์ผลการทดลองออกมาในรูปแผนภูมิที่แสดงแนวโน้มของตัวชี้วัดเปรียบเทียบกับการเพิ่มจำนวนหน่วยประมวลผลเป็นทวีคูณ จาก 1 เป็น 2 จาก 2 เป็น 4 จนกระทั่งถึง 8 หน่วยประมวลผล

3.2 การทดสอบเวลาการทำงานของฟังก์ชันในระบบโปรแกรม MapServer

ในขั้นตอนแรกของการดำเนินงานวิจัยนี้ จะเป็นส่วนของการทดสอบเวลาการทำงานของฟังก์ชันหลักๆ ในระบบโปรแกรม MapServer จุดมุ่งหมายของการดำเนินการในขั้นตอนนี้ก็คือเพื่อที่จะค้นหาฟังก์ชันงานที่เหมาะสมสำหรับใช้ในการทดลองของงานวิจัยมากที่สุด โดยฟังก์ชันดังกล่าวมีคุณสมบัติก็คือ ใช้เวลาการประมวลผลข้อมูลค่อนข้างมาก เวลาการตอบสนองต่อผู้ใช้งานช้า และมีการทำงานกับข้อมูลที่ขนาดรวมทั้งปริมาณค่อนข้างมาก ซึ่งในงานวิจัยนี้ ได้สร้างวิธีการทดสอบเวลาการทำงานของฟังก์ชันสำคัญๆ อย่างง่ายๆ ขึ้นมาเพื่อทดสอบว่า มีฟังก์ชันใดบ้างของระบบโปรแกรม MapServer ที่ตรงกับเงื่อนไขหรือคุณสมบัติที่กล่าวข้างต้น โดยรายงานผลการทดสอบที่จะนำเสนอในส่วนต่อไปนี้ จะเริ่มต้นนำเสนอที่ สภาพแวดล้อมที่ใช้ในการทดสอบ และจะนำเสนอชั้นข้อมูลที่ใช้เป็นชุดข้อมูล GIS ชุดหลักสำหรับการทดสอบ รวมไปถึงรายละเอียดวิธีการทดสอบ ในตอนท้ายนำเสนอผลลัพธ์เวลาการประมวลผลในแต่ละฟังก์ชันที่วัดได้ในการทดสอบ พร้อมทั้งการวิเคราะห์และสรุปผลการทดสอบ

3.2.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

1. หน่วยประมวลผล : Mobile Pentium4 1.66 GHz.[1]
2. ระบบปฏิบัติการ : Linux TLE 5.0 (Redhat 8.0)
3. โปรแกรมเว็บบราวเซอร์ (Browser): Mozilla 1.2.1
4. Web Server: Apache 2.0.47
5. ระบบโปรแกรมสารสนเทศทางภูมิศาสตร์ : Minnesota MapServer 4.0
6. ชุดข้อมูล GIS: แผนที่ประเทศไทย ความละเอียด 1:20000

ตารางที่ 3.1
ชั้นข้อมูลที่ใช้ในการทดสอบ

ชั้นข้อมูล	ชื่อชั้นข้อมูล	ประเภทชั้นข้อมูล	จำนวนรายการข้อมูล	ขนาด (MB)	คำอธิบาย
A	Admin_poly	Polygon	7988	46.5	ขอบเขตการปกครอง
B	Hydrology	Polygon	44537	220	แหล่งน้ำ
C	Admin_line	Line	22429	24.8	ขอบเขตการปกครอง
D	Railway	Line	436	0.7	ทางรถไฟ
E	L_trans	Line	388709	287	โครงข่ายถนน
F	Landmark	Point	34909	12.4	สถานที่สำคัญ

3.2.2 ขั้นตอนการทดสอบ

A และ B จะเป็นชั้นข้อมูลที่ใช้เป็นแผนที่ฐาน ทุกๆ การทดสอบจะมีทั้ง 2 ชั้นข้อมูลปรากฏอยู่เสมอ

---

[1] สภาพแวดล้อมที่ใช้สำหรับการทดสอบ เป็นคนละชุดกับส่วนการทดสอบประสิทธิภาพโปรแกรม MapServer แบบขนานที่พัฒนาขึ้นใหม่ เนื่องจากไม่มีผลต่อการคำนวณโดยรวม การทดสอบเวลาการทำงาน (Profile) เพื่อต้องการหาสัดส่วนของฟังก์ชันที่ใช้เวลานานที่สุดเท่านั้น

การทดสอบที่ 1 – ทดสอบการวาดแผนที่ฐานกับทุกชั้นข้อมูลที่ต้องการทดสอบ B D E และ F ที่ระบบ Full Extent (ระดับสูงสุดในการวาดแผนที่ประเทศไทย)

การทดสอบที่ 2 – ทดสอบฟังก์ชัน zoom in และ zoom out จากระดับ Full Extent ไปที่ระดับการ zoom ขนาดต่างๆ (2X 4X และ 8X)

การทดสอบที่ 3 – ทดสอบฟังก์ชันการ Query 3 แบบ

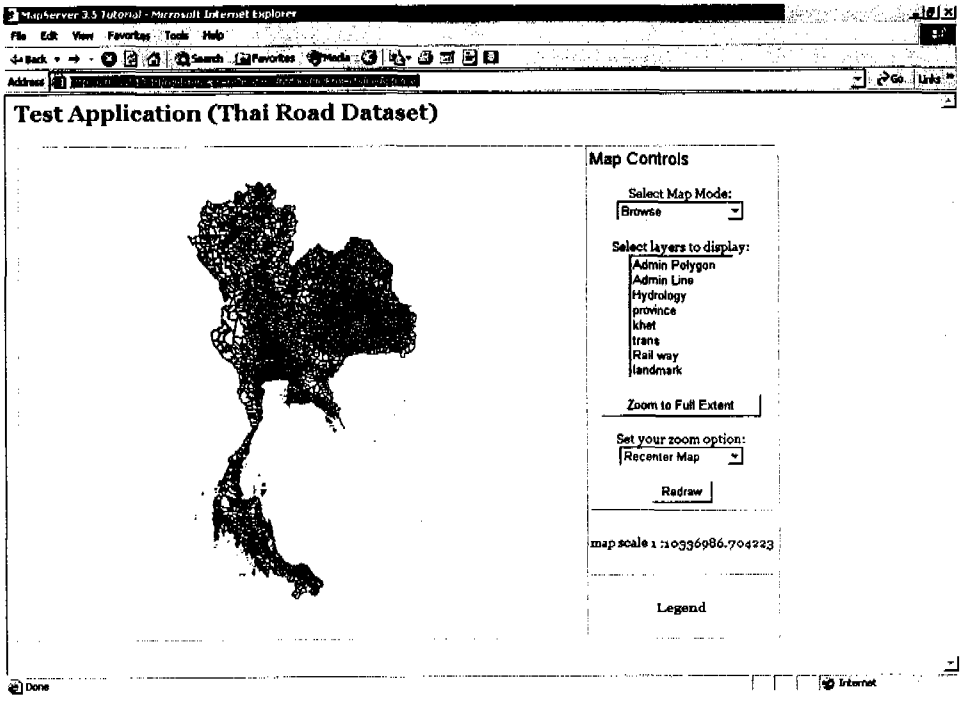
1. Point Query (Identify) – การสืบค้นข้อมูลโดยกำหนดจุดพิกัดอ้างอิงบนแผนที่

2. Shape Query – การสืบค้นโดยใช้กลุ่มของจุดพิกัดซึ่งประกอบกันเป็นพื้นที่รูปหลายเหลี่ยม (Polygon Area) ในการสืบค้นข้อมูล

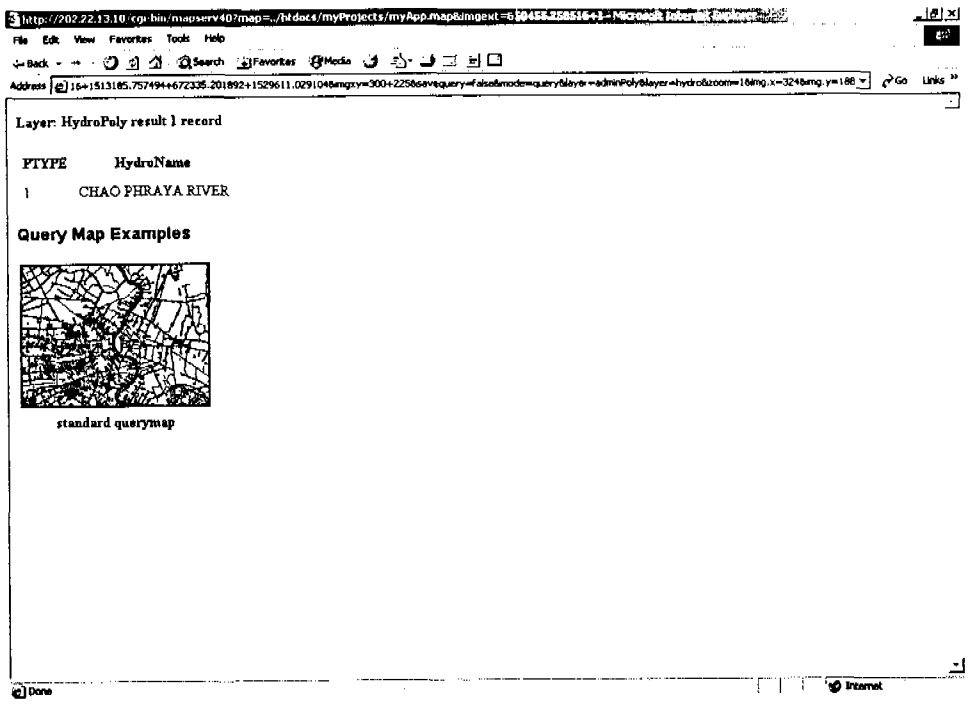
3. Feature Query (Spatial Search หรือ Overlay) – สืบค้นข้อมูลโดยใช้ Feature ของ ชั้นข้อมูลที่กำหนดไว้ ในการสืบค้นข้อมูลจากอีกชั้นข้อมูลหนึ่ง (โดยในระบบโปรแกรม MapServer กำหนดว่าชั้นข้อมูลที่กำหนดให้เป็นชั้นข้อมูลที่ใช้ในการสืบค้นจะต้องเป็น ชั้นข้อมูลชนิด Polygon เท่านั้น)

ภาพที่ 3.1

ตัวอย่างโปรแกรมประยุกต์ที่ใช้สำหรับการทดสอบ



ภาพที่ 3.2
หน้าจอที่เป็นผลลัพธ์ในการสืบค้นข้อมูล



3.2.3 ผลการทดสอบ

การทดสอบที่ 1 ทดสอบวัดเวลาการวาดแต่ละชั้นข้อมูลของแผนที่ฐานที่ระดับ Full-Extent หรือระดับสูงสุดในการแสดงแผนที่ประเทศไทย มาตราส่วน 1:10336986.704223 (เมตร) รายละเอียดผลการทดสอบแสดง ณ ตารางที่ 3.2

ตารางที่ 3.2

ผลการทดสอบที่ 1 ทดสอบเวลาการวาดชั้นข้อมูลแผนที่ฐาน

ชั้นข้อมูล	ชื่อชั้นข้อมูล	เวลาในการวาด (วินาที)
B	Hydrology	23
D	Railway	5
E	L_trans	17
F	Landmark	6

การทดสอบที่ 2 ทดสอบวัดเวลาการทำงานของฟังก์ชัน zoom in และ zoom out ณ ระดับ zoom ภาพแผนที่แตกต่างกันไป (2x 4x และ 8x) แสดงผลการทดสอบดังตารางที่ 3.3

ตารางที่ 3.3

ผลการทดสอบที่ 2 ทดสอบฟังก์ชัน zoom in และ zoom out

ชั้นข้อมูล	ชื่อชั้นข้อมูล	เวลาในการคำนวณ (วินาที)					
		Zoom in			Zoom out		
		2x	4x	8x	2x	4x	8x
B	Hydrology	35	29	22	26	31	39
D	Railway	6	4	2	2	3	4
E	L_trans	24	20	18	19	23	28
F	Landmark	4	2	1.5	2	3	4

การทดสอบที่ 3 ทดสอบวัดเวลาการทำงานของฟังก์ชันการสืบค้นข้อมูล 3 รูปแบบ โดยสำหรับฟังก์ชันการสืบค้นที่จะทำการทดสอบแบบแรก Point Query รูปแบบการทดสอบ จะวัดเวลาการสืบค้นเฉลี่ยสำหรับการสืบค้นข้อมูล 1 รายการแสดงเวลาที่ได้หลังจากทำการทดสอบดังตารางที่ 3.4

ตารางที่ 3.4

ผลการทดสอบที่ 3 การทดสอบฟังก์ชัน Point Query

ชั้นข้อมูล	ชื่อชั้นข้อมูล	ประเภทชั้นข้อมูล	จำนวนรายการผลลัพธ์	เวลาการสืบค้น (วินาที)
A	Admin_poly	Polygon	1	11
B	Hydrology	Polygon	1	40
D	Railway	Line	1	5
E	L_trans	Line	1	27
F	Landmark	Point	1	6

สำหรับการทดสอบฟังก์ชันการสืบค้นที่ 2 ฟังก์ชัน Shape Query จะแบ่งการทดสอบออกเป็น 2 ครั้ง โดยทั้งสองครั้งมีจุดมุ่งหมายในการวัดเวลาการสืบค้นข้อมูลของฟังก์ชันเหมือนกัน แต่จุดที่แตกต่างคือ ครั้งแรกจะสืบค้นโดยใช้พื้นที่การสืบค้นขนาดครอบคลุมขอบเขตจังหวัดกรุงเทพฯ ส่วนครั้งที่ 2 จะใช้พื้นที่การสืบค้นขนาดใหญ่กว่าครั้งแรก นั่นคือ ใช้พื้นที่การสืบค้นขนาดครอบคลุมพื้นที่ภาคกลางตอนล่าง (ตั้งแต่ชัยนาท/ลพบุรี ลงมาจนถึงสมุทรปราการ) แสดงผลการทดสอบฟังก์ชัน Shape Query ดังตารางที่ 3.5 และตารางที่ 3.6

ตารางที่ 3.5

ผลการทดสอบที่ 3 การทดสอบฟังก์ชัน Shape Query ใช้พื้นที่การสืบค้นครอบคลุมขอบเขตจังหวัดกรุงเทพฯ

ชั้นข้อมูล	ชื่อชั้นข้อมูล	ประเภทชั้นข้อมูล	จำนวนรายการผลลัพธ์	เวลาการสืบค้น (วินาที)
A	Admin_poly	Polygon	258	9
B	Hydrology	Polygon	4051	56
D	Railway	Line	116	6
E	L_trans	Line	8990	65 (1.05 นาที)
F	Landmark	Point	11283	28

ตารางที่ 3.6

ผลการทดสอบที่ 3 การทดสอบฟังก์ชัน Shape Query ใช้พื้นที่การสืบค้นเขตภาคกลางตอนล่าง

ชั้นข้อมูล	ชื่อชั้นข้อมูล	ประเภทชั้นข้อมูล	จำนวนรายการผลลัพธ์	เวลาการสืบค้น (วินาที)
A	Admin_poly	Polygon	1602	16
B	Hydrology	Polygon	10113	71 (1.11 นาที)
D	Railway	Line	265	7
E	L_trans	Line	92874	270 (4.30 นาที)
F	Landmark	Point	16761	42

การทดสอบฟังก์ชันการสืบค้นฟังก์ชันสุดท้าย ฟังก์ชัน Feature Query รูปแบบการทดสอบแบ่งออกเป็น 2 ครั้ง โดยครั้งแรกจะเป็นการทดสอบวัดเวลาการสืบค้นของฟังก์ชัน Feature Query ในแบบที่กำหนดให้ชั้นข้อมูลที่ถูกสืบค้นมีได้เพียงชั้นข้อมูลเดียว สำหรับในครั้งที่ 2 ของการทดสอบจะทดสอบในรูปแบบให้ชั้นข้อมูลที่ถูกสืบค้นสามารถมีได้มากกว่าหนึ่งชั้นข้อมูลซึ่งในการทดสอบจะทดสอบที่ 2 และ 3 ชั้นข้อมูลตามลำดับ ในส่วนการใช้สัญลักษณ์สำหรับอธิบายผลการทดสอบ จะใช้สัญลักษณ์ในรูปแบบ layer1 -> layer2 แทน ชั้นข้อมูลตัวตั้งที่ใช้ในการสืบค้น -> ชั้นข้อมูลที่ถูกสืบค้นข้อมูล แสดงผลการทดสอบฟังก์ชัน Feature Query ดัง ตารางที่ 3.7 และตารางที่ 3.8

ตารางที่ 3.7

ผลการทดสอบที่ 3 การทดสอบฟังก์ชัน Feature Query (ชั้นข้อมูลที่ถูกสืบค้น 1 ชั้นข้อมูล)

ชั้นข้อมูล	ประเภทชั้นข้อมูล	Feature ตัวตั้ง	จำนวนรายการผลลัพธ์	เวลาการสืบค้น (วินาที)
A -> B	Polygon -> Polygon	จังหวัด กรุงเทพมหานคร	3008	67 (1.07 นาที)
		เขตพระนคร จังหวัดกรุงเทพ	7	39
		ตำบลท่าข้าม บางขุนเทียน	215	53

ตารางที่ 3.7

ผลการทดสอบที่ 3 การทดสอบฟังก์ชัน Feature Query (ชั้นข้อมูลที่ถูกสืบค้น 1 ชั้นข้อมูล) (ต่อ)

ชั้นข้อมูล	ประเภทชั้นข้อมูล	Feature ตัวตั้ง	จำนวนรายการผลลัพธ์	เวลาการสืบค้น (วินาที)
A -> B	Polygon -> Polygon	ตำบลบางขุนเทียน จอมทอง	62	40
		ตำบลบางโคล่ บางคอแหลม	17	40
		ตำบลปากน้ำ อ.เมือง สมุทรปราการ	7	37
		ตำบล ชนะสงคราม เขต พระนคร	2	35
B -> A	Polygon -> Polygon	แม่น้ำชี (ความยาว 765 km)	-	มากกว่า 15 นาที (Server timeout)
		แม่น้ำยม (ความยาวระหว่าง 365 ถึง 765)	95	295 (4.55 นาที)
		แม่น้ำเจ้าพระยา (365 km)	215	155 (2.35 นาที)
		แม่น้ำปิง	55	58
		แม่น้ำป่าสัก	55	37
		แม่น้ำลพบุรี	34	33
		บึงบอระเพ็ด นครสวรรค์	7	35
A -> E	Polygon -> Line	กรุงเทพมหานคร	5575	63 (1.03 นาที)
		เขตพระนคร	322	31
		ตำบล ชนะสงคราม เขตพระนคร	8	27
A -> F	Polygon -> Point	จังหวัด กรุงเทพมหานคร	8230	55
		เขตพระนคร จังหวัดกรุงเทพ	273	9
		ตำบล ชนะสงคราม เขตพระนคร	19	2

ตารางที่ 3.8

ผลการทดสอบที่ 3 การทดสอบฟังก์ชัน Feature Query

(ชั้นข้อมูลที่ถูกสืบค้นมากกว่า 1 ชั้นข้อมูล)

ชั้นข้อมูล	ประเภทชั้นข้อมูล	Feature ตัวตั้ง	จำนวนรายการผลลัพธ์	เวลาการสืบค้น (วินาที)
A -> B, F	Poly -> poly, point	กรุงเทพ	11238	87 (1.27 นาที)
A -> B, E	Poly -> poly, line	กรุงเทพ	8583	112 (1.52 นาที)
A -> B, E, F	Poly -> poly, line, point	กรุงเทพ	16813	132 (2.12 นาที)

3.2.4 วิเคราะห์ผลการทดสอบ

จากผลทดสอบสามารถสรุปเป็นข้อๆ ได้ดังนี้

1. ชั้นข้อมูลประเภท Polygon เช่น Hydrology จะเป็นชั้นข้อมูลที่ใช้เวลาการทำงานใน แต่ละฟังก์ชันมากที่สุด ทั้งนี้เนื่องมาจาก Polygon เป็น Feature ที่ประกอบด้วยจุดมากที่สุด อีกทั้งขนาดของไฟล์ข้อมูลก็มีขนาดใหญ่มาก จึงทำให้ใช้เวลาในการคำนวณและประมวลผลเป็นเวลานานมากกว่าชั้นข้อมูลประเภทอื่น

2. ฟังก์ชันที่ใช้เวลาการประมวลผลมากที่สุด ได้แก่ Feature Query โดยเวลาการทำงานโดยทั่วไปจะมากกว่า 30 วินาทีขึ้นไป และ ปัจจัยที่มีผลต่อเวลาการทำงานของฟังก์ชัน Feature Query ก็คือ

ก.) ชนิดของ Feature ที่ถูกสืบค้น (Query) ถ้า Feature เป็นข้อมูลประเภท Polygon จะใช้เวลาการสืบค้นข้อมูลมากที่สุด (Polygon > line > point)

ข.) ขนาดของข้อมูล Polygon ที่นำมาใช้เป็น Feature ตัวตั้ง ในการสืบค้นข้อมูล และ ขนาดของ Feature ที่เป็นชั้นข้อมูลที่ถูกสืบค้น

ค.) จำนวนรายการที่ถูกสืบค้นได้หรือรายการที่เป็นผลลัพธ์ทั้งหมดของการสืบค้น

3. ฟังก์ชันที่ใช้เวลาการทำงานเป็นลำดับที่ 2 คือ Shape Query ซึ่งเวลาการทำงานขึ้นอยู่กับขนาดของพื้นที่ที่ใช้ในการสืบค้นข้อมูล ถ้าขนาดพื้นที่ใหญ่ จะทำให้ครอบคลุม Feature ที่ถูกสืบค้นได้มาก เวลาการทำงานก็จะมากตามไปด้วย

4. ข้อสังเกต ที่ขนาดมาตรส่วนแผนที่ (Scale) สูงๆ เช่น ระดับที่มองเห็นประเทศไทยทั้งประเทศ การแสดงผลข้อมูลจะใช้เวลานาน ทั้งนี้เนื่องมาจากว่า ในระดับมาตราส่วนสูงๆ จะทำให้มองเห็น Feature ของแต่ละ ชั้นข้อมูลจำนวนมาก ทำให้เวลาที่ระบบใช้ในการวาดข้อมูล ใช้เวลานานกว่าการวาดข้อมูลที่ขนาดมาตราส่วนต่ำๆ

3.2.5 สรุปผลการทดสอบ

ฟังก์ชัน Feature Query เป็นฟังก์ชันที่มีคุณสมบัติ ตรงกับเงื่อนไขที่งานวิจัยค้นหาสามารถนำมาใช้ในงานวิจัยได้ โดยปัจจัยที่ต้องคำนึงก็คือ ประเภทข้อมูล ขนาดของไฟล์ข้อมูล รวมทั้งขนาดของ Feature ที่นำมาใช้งานในฟังก์ชันนี้ ถ้ามีขนาดใหญ่ ย่อมมีผลต่อเวลาและประสิทธิภาพการทำงานของฟังก์ชัน จากการทดสอบจะเห็นได้ชัดเกี่ยวกับเรื่องเวลาการทำงานของฟังก์ชันที่ใช้เวลามากพอสมควร ยิ่งถ้าชั้นข้อมูลที่ถูกสืบค้นมีมากกว่า 1 ชั้นข้อมูลเวลาการทำงานก็จะมากยิ่งขึ้นตามไปด้วย

3.3 การตรวจสอบขั้นตอนการทำงานของระบบโปรแกรม MapServer

วิธีการตรวจสอบขั้นตอนการทำงานของระบบโปรแกรม MapServer นั้น งานวิจัยนี้เลือกใช้เครื่องมือ gprof ซึ่งเป็นเครื่องมือ Profiling ที่เป็นมาตรฐานบนระบบปฏิบัติการลินุกซ์ สำหรับการตรวจสอบการทำงานของโปรแกรมและเวลาการทำงานของแต่ละฟังก์ชัน ซึ่งผลลัพธ์ที่ได้จากการใช้เครื่องมือ gprof จะแบ่งได้ออกเป็น 2 ลักษณะ ผลลัพธ์แรก Flat profile จะบอกรายละเอียดเกี่ยวกับเวลาที่แต่ละฟังก์ชันในโปรแกรมใช้ในการประมวลผล และจำนวนครั้งที่ฟังก์ชันแต่ละฟังก์ชันถูกเรียกใช้ ส่วนผลลัพธ์ที่สอง จะเป็นส่วนของการแสดง call graph หรือแผนผังลำดับการเรียกใช้งานฟังก์ชัน สามารถใช้ศึกษาความสัมพันธ์ และลำดับการเรียกทำงานของแต่ละฟังก์ชันในระบบโปรแกรม MapServer

ภาพที่ 3.3

ภาพผลลัพธ์ของ Flat profile ที่ได้จากการใช้เครื่องมือ gprof

Flat profile:

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
83.35	56.43	56.43	1528305187	0.00	0.00	msIntersectSegments
16.13	67.35	10.92	865	0.01	0.08	msIntersectPolygons
0.41	67.63	0.28	18885	0.00	0.00	msPointInPolygon
0.01	67.64	0.01	7408	0.00	0.00	msEncodeUrl
0.01	67.65	0.01	1636	0.00	0.00	msSHPReadShape
0.01	67.66	0.01	436	0.00	0.00	processLine
0.01	67.67	0.01	431	0.00	0.00	msTransformShapeToPixel
0.01	67.68	0.01	416	0.00	0.00	msClipPolygonRect
0.01	67.69	0.01	294	0.00	0.00	imageFilledPolygon
0.01	67.70	0.01	10	0.00	0.00	msSHPOpen
0.00	67.70	0.00	554132	0.00	0.00	SwapWord
0.00	67.70	0.00	127489	0.00	0.00	msSHPReadBounds
0.00	67.70	0.00	127484	0.00	0.00	msRectOverlap
0.00	67.70	0.00	127479	0.00	0.00	msGetBit
0.00	67.70	0.00	88231	0.00	0.00	gsub
0.00	67.70	0.00	10325	0.00	0.00	msIntersectPointPolygon
0.00	67.70	0.00	6512	0.00	0.00	imageScanline
0.00	67.70	0.00	4583	0.00	0.00	msDBFReadAttribute
0.00	67.70	0.00	4583	0.00	0.00	msDBFReadStringAttribute
0.00	67.70	0.00	3697	0.00	0.00	msInitShape
0.00	67.70	0.00	3494	0.00	0.00	isOn
0.00	67.70	0.00	2180	0.00	0.00	clipLine
0.00	67.70	0.00	2127	0.00	0.00	msEvalExpression
0.00	67.70	0.00	1636	0.00	0.00	msFreeShape
0.00	67.70	0.00	1495	0.00	0.00	msFree
0.00	67.70	0.00	1208	0.00	0.00	msLayerNextShape
0.00	67.70	0.00	1203	0.00	0.00	msSetBit
0.00	67.70	0.00	1203	0.00	0.00	msShapeGetClass

จากภาพที่ 3.3 ผลลัพธ์ของ Flat profile จากเครื่องมือ gprof ที่ได้จากการทำงานของโปรแกรม MapServer แสดงให้เห็นว่า ฟังก์ชันที่ใช้เวลามากที่สุด คือ ฟังก์ชัน msIntersectSegments ใช้เวลาการคำนวณถึง 83.35% และรองลงมาคือ ฟังก์ชัน msIntersectPolygon ใช้เวลา 16.13% ของเวลาที่โปรแกรมทำงานทั้งหมด ซึ่งจากผลลัพธ์ที่ได้สามารถตรวจสอบการทำงานของ MapServer ได้ ในแง่ที่ว่าฟังก์ชันส่วนใดของโปรแกรม MapServer ในส่วนฟังก์ชันงาน FeatureQuery ที่ใช้เวลาการประมวลผลมาก และควรเน้นศึกษาเพื่อปรับปรุงประสิทธิภาพการทำงานเป็นพิเศษ

ภาพที่ 3.4

ภาพผลลัพธ์ของ Call graph ที่ได้จากการใช้เครื่องมือ gprof

```
                     Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) for 0.01% of 67.70 seconds

index % time    self  children    called     name
                                               <spontaneous>
[1]    100.0    0.00   67.70                 main [1]
                0.00   67.64       1/1           msQueryByFeatures [2]
                0.00    0.06       1/1           msReturnTemplateQuery [7]
                0.00    0.00       1/1           msQueryByPoint [28]
                0.00    0.00       6/3494        isOn [39]
                0.00    0.00       2/2           msGetLayerIndex [124]
                0.00    0.00       1/1           msAllocMapServObj [153]
                0.00    0.00       1/1           loadParams [150]
                0.00    0.00       1/1           loadMap [148]
                0.00    0.00       1/1           msOWSDispatch [174]
                0.00    0.00       1/1           loadForm [147]
                0.00    0.00       1/1           msSaveQuery [179]
                0.00    0.00       1/1           writeLog [184]
                0.00    0.00       1/1           msFreeMapServObj [165]
                0.00    0.00       1/1           setCoordinate [183]
                0.00    0.00       1/5           msAdjustExtent [104]
-----------------------------------------------
                0.00   67.64       1/1           main [1]
[2]     99.9    0.00   67.64       1         msQueryByFeatures [2]
               10.92   56.71     865/865         msIntersectPolygons [3]
                0.00    0.01     866/1208        msLayerNextShape [26]
                0.00    0.00       2/10          msLayerOpen [23]
                0.00    0.00       1/433         msLayerGetShape [27]
                0.00    0.00     868/3697        msInitShape [38]
                0.00    0.00     866/1636        msFreeShape [42]
                0.00    0.00     865/1203        msShapeGetClass [45]
                0.00    0.00     215/216         addResult [58]
                0.00    0.00     214/214         msMergeRect [59]
                0.00    0.00       2/10          msLayerClose [88]
                0.00    0.00       1/2           msIsLayerQueryable [126]
                0.00    0.00       1/7           msLayerWhichItems [103]
```

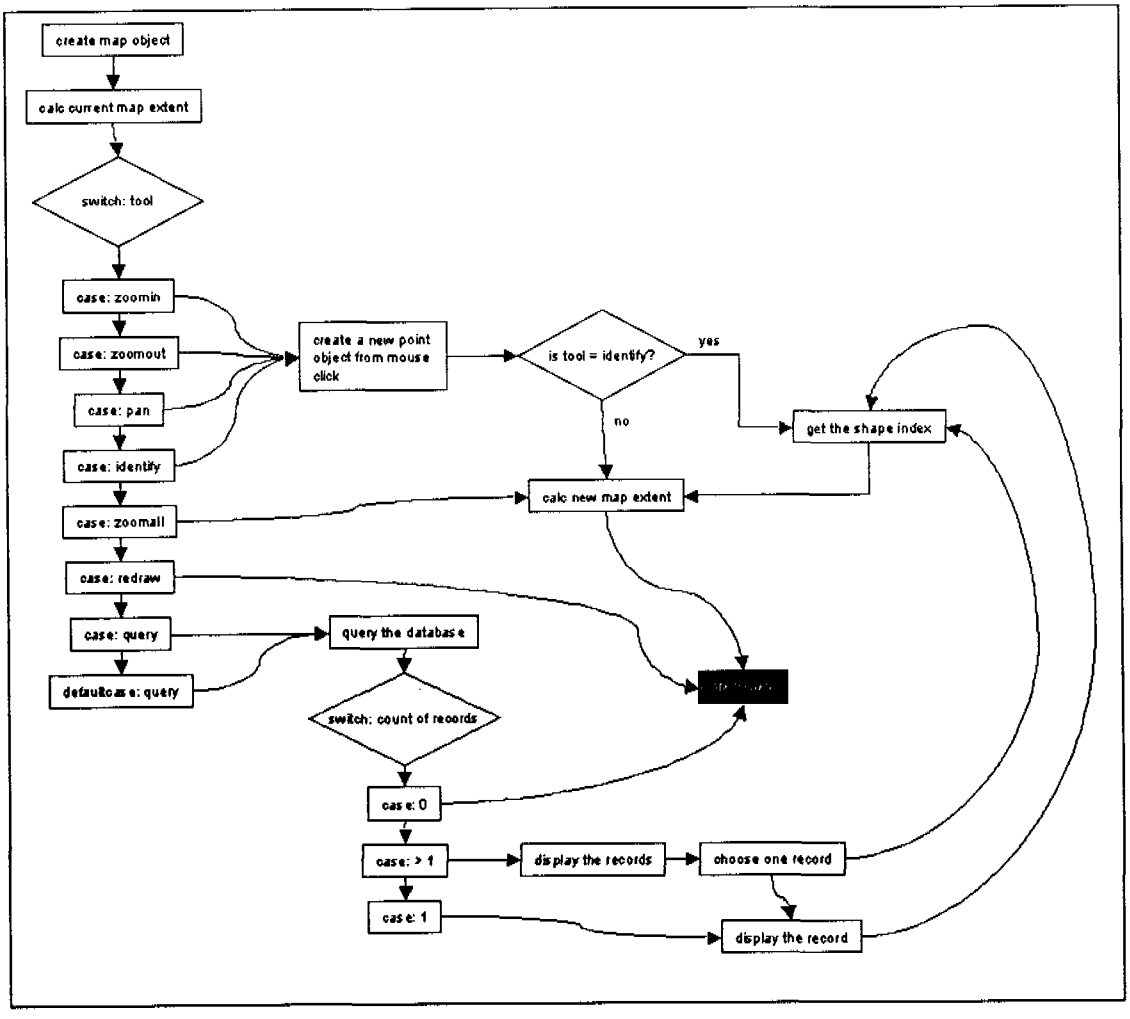
จากภาพที่ 3.4 ผลลัพธ์ของ Call graph จากเครื่องมือ gprof ที่ได้จากการทำงานของโปรแกรม MapServer จะแสดงลำดับการทำงาน ความสัมพันธ์กันของลำดับการเรียกใช้ฟังก์ชันที่เกิดขึ้นระหว่างโปรแกรม MapServer ทำงาน โดยจากที่แสดงในรูปจะเห็นได้ว่า การทำงานจะเริ่มที่ฟังก์ชัน Main เป็นฟังก์ชันแรก ซึ่งเวลาการทำงานที่ใช้ทั้งหมด 67.70 วินาที และฟังก์ชัน Main จะมีการเรียกให้ฟังก์ชัน msQueryByFeatures ทำงานเป็นลำดับต่อไป โดยที่โปรแกรมใช้เวลาการ

ทำงานอยู่ในฟังก์ชันนี้เป็นเวลาทั้งหมด 67.64 วินาที จากนั้นเมื่อ msQueryByFeatures ทำงานเสร็จสิ้นก็จะเรียกใช้ฟังก์ชัน msReturnTemplateQuery และเรียกฟังก์ชันอื่น ต่อไปเรื่อยๆ ตามลำดับที่แสดงไว้ใน Call graph

หลังจากที่ใช้เครื่องมือ gprof ตรวจสอบการทำงานของระบบโปรแกรม MapServer แล้ว ลำดับต่อไปก็จะเริ่มทำการตรวจสอบซอร์สโค้ดและลำดับการทำงานของระบบโปรแกรม MapServer โดยใช้เครื่องมือมาตรฐานบนระบบปฏิบัติการลินุกซ์ อีกเครื่องมือหนึ่ง นั่นคือ GDB ซึ่งในงานวิจัยนี้ GDB ถูกใช้เป็นเครื่องมือช่วยในการศึกษาการทำงานในระดับซอร์สโค้ด ว่าระบบโปรแกรม MapServer มีการทำงานภายในจริงๆ เป็นเช่นไร รวมทั้งศึกษาว่าระบบโปรแกรม MapServer มีวิธีการจัดการกับข้อมูลอย่างไร ให้ได้ผลลัพธ์ตามที่ผู้ใช้งานต้องการ เมื่อตรวจสอบการทำงานของระบบโปรแกรม MapServer ในส่วนฟังก์ชันที่สนใจเสร็จเรียบร้อยแล้ว จะทำการจัดสร้าง flow chart เพื่อแสดงลำดับการทำงานของโปรแกรม MapServer ไว้ใช้สำหรับอ้างอิงในส่วนการพัฒนาต่อไป ซึ่งในการตรวจสอบขั้นตอนการทำงานของโปรแกรม MapServer สามารถใช้ประโยชน์ในการหาจุดของอัลกอริทึมที่จะปรับเปลี่ยนรูปแบบการพัฒนาไปเป็นการพัฒนาโปรแกรมในแนวทางการประมวลผลแบบขนานได้ อีกทั้งในส่วนการวิเคราะห์โปรแกรมโดยใช้เครื่อง gprof ยังสามารถหาส่วนของฟังก์ชันย่อยที่ใช้เวลาการทำงานค่อนข้างมาก ซึ่งส่วนของฟังก์ชันย่อยส่วนนี้ น่าจะเป็นส่วนที่นำมาปรับปรุงประสิทธิภาพมากที่สุด

ภาพที่ 3.5

ลำดับขั้นตอนหลักในการทำงานของระบบโปรแกรม MapServer



ที่มา: โฮมเพจ http://mapserver.gis.umn.edu/

ขั้นตอนของการทำงานหลักของระบบโปรแกรม MapServer จะเริ่มขึ้นเมื่อมี Request จากผู้ใช้งานเข้ามา ขั้นตอนแรกโปรแกรมจะสร้าง Map object ขึ้นมาเพื่อใช้สำหรับจัดเก็บข้อมูลและแลกเปลี่ยนข้อมูลสำคัญระหว่างการประมวลผล จากนั้นขั้นตอนต่อมาจะนำข้อมูลจาก Request ของผู้ใช้งานมาคำนวณว่าตำแหน่งพิกัดปัจจุบันของแผนที่อยู่ที่ตำแหน่งใด หลังจากคำนวณพิกัดและทราบตำแหน่งของพิกัดปัจจุบัน รวมทั้งขนาดมาตราส่วนของแผนที่ (map scale) แล้ว โปรแกรมจะดำเนินการในส่วนของการตรวจสอบว่า request ที่รับมาจากผู้ใช้งาน เป็น request ของการทำงานที่ฟังก์ชันหรือเครื่องมือใด ซึ่งตรวจสอบโดยใช้คำสั่ง switch-case เมื่อ

ตรวจสอบแล้วก็จะเข้าไปทำงานและประมวลผลตามการทำงานของฟังก์ชันหรือเครื่องมือนั้นๆ ตัวอย่างเช่น ถ้าเป็นฟังก์ชันในกลุ่มฟังก์ชันการเรียกแสดงรูปแผนที่ (Navigation function) อาทิ ฟังก์ชันย่อ/ขยายรูปแผนที่ (Zoom) ฟังก์ชันเลื่อนภาพแผนที่ (Pan) การทำงานหลักๆ ของฟังก์ชันกลุ่มนี้คือ เริ่มแรกจะทำการสร้าง point object เพื่อจัดเก็บจุดที่ผู้ใช้งานคลิกบนแผนที่ไว้ใช้ในการคำนวณ จากนั้นจึงคำนวณกรอบพิกัดของแผนที่ใหม่ (extent) ตามจุดที่ผู้ใช้งานระบุ เมื่อคำนวณเสร็จแล้วจึงค่อยวาดรูปแผนที่ใหม่ เพื่อนำมาแสดงผลตามที่ผู้ใช้งานต้องการอีกครั้ง หรืออีกตัวอย่างหนึ่งถ้าเป็นฟังก์ชันการสืบค้นข้อมูล (Query function) การทำงานก็คือ นำข้อมูลรายละเอียดที่ผู้ใช้งานระบุ เข้าไปสืบค้นยังฐานข้อมูล เมื่อสืบค้นข้อมูลเสร็จ จะมาทำงานในส่วนของการตรวจสอบว่าค้นพบข้อมูลกี่รายการ ซึ่งถ้าค้นพบข้อมูลตั้งแต่ 1 รายการขึ้นไป โปรแกรมก็จะแสดงผลรายการที่ค้นพบทั้งหมดพร้อมทั้งแสดงแผนที่ที่ใช้ในการสืบค้น แต่ถ้าในการสืบค้นไม่พบรายการข้อมูลก็จะแสดงผลที่ฝั่งผู้ใช้งานเฉพาะภาพแผนที่เท่านั้น

3.4 การศึกษาความสัมพันธ์ในการใช้งานข้อมูล (Data Dependency Analysis)

เพื่อให้สามารถแน่ใจได้ว่า สามารถประยุกต์นำเอาเทคนิคการพัฒนาโปรแกรมแบบขนานมาใช้แก้ปัญหาได้ ขั้นตอนการศึกษาความสัมพันธ์ในการใช้ข้อมูลของโปรแกรม (Data Dependency Analysis) จึงเป็นขั้นตอนที่สำคัญมากขั้นตอนหนึ่งในงานวิจัยก่อนเริ่มต้นพัฒนาโปรแกรม

Dependency หรือความเกี่ยวพันธ์กันภายในโปรแกรม เกิดจาก การที่ลำดับการเรียกใช้ประโยคคำสั่งแต่ละประโยคในโปรแกรมมีผลต่อผลลัพธ์ที่เกิดขึ้นจากการประมวลผลของโปรแกรม Data Dependency เป็นประเภทของ Dependency อย่างหนึ่ง ซึ่งมีส่วนสำคัญอย่างยิ่งในการแปลงโปรแกรมจากโปรแกรมแบบลำดับให้เป็นโปรแกรมแบบขนาน เกิดจากการที่ประโยคคำสั่งในโปรแกรม มีการเรียกใช้หน่วยความจำที่เดียวกันหลายๆ ครั้ง หรือประโยคคำสั่งที่ในโปรแกรมมีการอ้างถึงตัวแปร (หน่วยความจำ) ที่ต้องเกิดจากการคำนวณในประโยคก่อนหน้า ทำให้เกิดความสัมพันธ์ในการใช้ข้อมูลระหว่างประโยคเกิดขึ้น ตัวอย่างของ Data Dependency แสดงดังภาพที่ 3.6

ภาพที่ 3.6

ตัวอย่างโปรแกรมที่เกิด Data Dependency

```
a = b + 1;          (1)
c = 4 – a;          (2)
b = 2a – c;         (3)
```

จากภาพที่ 3.6 สามารถแสดงให้เห็นถึง Data Dependency ที่เกิดขึ้นระหว่างประโยคคำสั่ง โดยแต่ละประโยคต้องการค่าของตัวแปรที่เป็นผลลัพธ์ที่ได้จากประโยคอื่น เช่น ประโยคที่ 1 ต้องการค่าของ b ซึ่งเป็นผลลัพธ์ของประโยคที่ 3 เป็นต้น นอกจาก Data Dependency จากตัวอย่างข้างต้นแล้ว Data Dependency ที่สำคัญอีกรูปแบบหนึ่ง ซึ่งสามารถเกิดขึ้นได้จากการใช้งานคำสั่ง ควบคุมการทำงานของโปรแกรม (Flow Control Statement) ประเภท loop มีชื่อเรียกว่า loop-carried dependency ตัวอย่างของ loop-carried dependency แสดงดังภาพที่ 3.7

ภาพที่ 3.7

ตัวอย่างโปรแกรมที่เกิด loop-carried Dependency

```
for (j=1; j < 5; j++){
    A[j] = A[j – 1]  *  2;
}
```

จากภาพที่ 3.7 แสดงให้เห็นตัวอย่างโปรแกรมที่เกิด loop-carried dependency โดยในการที่จะคำนวณได้ผลลัพธ์ค่า A[j] ออกมานั้นจะต้องได้ผลลัพธ์ของ A[j-1] ซึ่งเป็นผลลัพธ์ของ loop รอบก่อนหน้าด้วย ดังนั้นทุกๆ รอบของ loop ในตัวอย่างจะมีความสัมพันธ์กับผลลัพธ์ของ loop รอบก่อนหน้าเสมอ เกิดเป็นความสัมพันธ์ data dependency ระหว่างรอบของ loop เกิดขึ้น

สำหรับวิธีในการศึกษาความสัมพันธ์ในการใช้ข้อมูลของโปรแกรมที่งานวิจัยนี้เลือกใช้คือการใช้เครื่องมือ GDB (GNU Debugger) ตรวจสอบลำดับการทำงานและความสัมพันธ์

ระหว่างค่าต่างๆ ที่เกิดขึ้นจากการคำนวณภายในโค้ดโปรแกรม MapServer โดยในการตรวจสอบจะใช้คำสั่งในเครื่องมือ GDB ตรวจสอบไปตามลำดับการทำงานของโปรแกรม โดยจะเน้นเป็นพิเศษสำหรับการตรวจสอบ loop-carried Dependency ระหว่างรอบการทำงานของคำสั่งควบคุมการทำงาน (loop) ทั้งนี้เนื่องจากจุดหลักที่งานวิจัยนี้จะใช้ในการปรับปรุงโปรแกรม MapServer ให้ประยุกต์ใช้เทคนิคการประมวลผลแบบขนานจะอยู่ที่ loop ขนาดใหญ่ในโปรแกรมนั่นเอง

ซึ่งโดยสรุปจากการตรวจสอบและศึกษาความสัมพันธ์ในการใช้ข้อมูล (Data Dependency) ในการดำเนินงานของงานวิจัยนี้ ไม่พบจุดที่เกิด Data Dependency ที่เป็นปัญหาสำคัญสำหรับการประยุกต์ใช้การประมวลผลแบบขนาน ในส่วนของฟังก์ชัน FeatureQuery ของโปรแกรม MapServer เลย

3.5 โครงสร้างระบบ

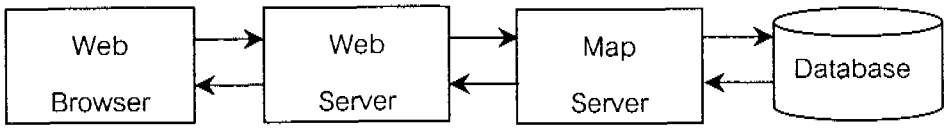
3.5.1 สถาปัตยกรรม 4-Tiered GIS web service

งานวิจัยนี้ออกแบบและพัฒนาโดยอ้างอิงกับโครงสร้างสถาปัตยกรรมระบบแบบ 4-Tiered GIS web service โดยโครงสร้างสถาปัตยกรรมระบบรูปแบบนี้ มีพื้นฐานมาจากโครงสร้างสถาปัตยกรรมระบบแบบ Multi-tier web application architecture ซึ่งเป็นโครงสร้างสถาปัตยกรรมโปรแกรมประยุกต์ในรูปแบบ Web-Based Application ที่ได้รับการยอมรับและถูกนำไปประยุกต์ใช้อย่างแพร่หลายในปัจจุบัน จุดเด่นของโครงสร้างสถาปัตยกรรมระบบแบบ 4-Tiered GIS web service ก็คือ ระบบมีความยืดหยุ่นค่อนข้างสูง สามารถปรับแต่งหรือปรับปรุงประสิทธิภาพของระบบโดยเพิ่มหรือลดส่วนประกอบของระบบได้ง่าย ระบบมีประสิทธิภาพในการประมวลผลสูง เนื่องมาจากมีการแบ่งแยกส่วนประกอบของระบบออกจากกันทำให้แต่ละส่วนสามารถทำงานบนฮาร์ดแวร์ของตนเองได้อย่างเต็มประสิทธิภาพ จากจุดเด่นดังกล่าวทำให้โครงสร้างสถาปัตยกรรมระบบแบบ 4-Tiered GIS web service ถูกนำไปใช้ในงานวิจัย Web-Based GIS หลายๆ งานวิจัย เช่น ในงานวิจัยของ (Tu, 2001) (Vatsavai, 2000) (Chen, 2000) และ (Pinet, 2000) เป็นต้น นอกจากนั้นยังมีการนำไปประยุกต์ใช้สำหรับการพัฒนาโปรแกรมประยุกต์ Web-Based GIS ในส่วนของซอฟต์แวร์เชิงพาณิชย์อีกด้วย ตัวอย่างเช่น โปรแกรมประยุกต์ ArcIMS และ MapServer โดยที่โครงสร้างระบบในรูปแบบนี้ จะแบ่งส่วนประกอบของระบบออกเป็น 4 ส่วน ได้แก่

1. เว็บบราวเซอร์ (Web Browser)
2. เว็บเซิร์ฟเวอร์ (Web Server)
3. แอพลิเคชันเซิร์ฟเวอร์หรือเครื่องแม่ข่ายสารสนเทศทางภูมิศาสตร์ (Map Server)
4. ฐานข้อมูล (Database)

ภาพที่ 3.8

4-Tiered GIS web service system



การทำงานของโครงสร้างระบบในแบบ 4-Tiered GIS web service จะเริ่มต้นขึ้นเมื่อผู้ใช้งานทำงานกับฟังก์ชันที่ต้องการบนเว็บบราวเซอร์ (Web Browser) เว็บบราวเซอร์จะทำการส่งคำร้องขอการทำงาน (Request) เข้าสู่ระบบ จากนั้น Web Server จะรับ Request ที่เกิดจากผู้ใช้งานเข้ามา ถ้ากรณีที่เป็น Request เกี่ยวกับการร้องขอหน้า HTML Web Server จะให้บริการด้วยตนเอง แต่ถ้าในกรณีที่เป็น Request จากผู้ใช้งานที่เกี่ยวกับการทำงานกับแผนที่ Web Server จะส่ง Request ต่อไปที่ Map Server เพื่อทำการประมวลผล Map Server เมื่อรับ Request เข้ามาแล้วจะไปดึงข้อมูลเชิงพื้นที่ ที่ใช้สำหรับ Request นั้นๆ เข้ามาประมวลผล จากนั้นจึงส่งข้อมูลผลลัพธ์ที่ได้กลับไปยัง Web Server และ Web Server ก็จะส่งผลลัพธ์ดังกล่าวกลับไปแสดงผลที่ฝั่ง Web Browser ผู้ใช้งานในรูปแบบของ HTML อีกครั้งหนึ่ง

3.5.2 เครื่องมือและซอฟต์แวร์หลักที่ใช้ในระบบ

เมื่อพิจารณาองค์ประกอบและโครงสร้างการทำงานของระบบข้างต้นแล้ว งานวิจัยนี้เลือกใช้ซอฟต์แวร์และเครื่องมือในการพัฒนาระบบดังแสดง ณ ตารางที่ 3.9

ตารางที่ 3.9

รายการเครื่องมือและซอฟต์แวร์หลักที่ใช้ในระบบ

เครื่องมือ	คำอธิบาย
1. Apache HTTP Server	ใช้เป็น Web Server รองรับ Request ที่เกิดจากผู้ใช้งานระบบ
2. Minnesota MapServer	ใช้เป็น Map Server ประมวลผลงานที่เกี่ยวข้องกับแผนที่และข้อมูลสารสนเทศทางภูมิศาสตร์
3. MPICH	ไลบรารีที่ใช้ในการเขียนโปรแกรมแบบขนาน โดยใช้เทคนิคการส่งผ่านข้อความ (Message Passing)
4. Linux	ระบบปฏิบัติการใช้สำหรับการทำงาน และการสร้างระบบคลัสเตอร์
5. แบวูฟล์คลัสเตอร์	ระบบคอมพิวเตอร์แบบขนานที่ใช้ในงานวิจัย
6. ข้อมูลแผน (ESRI Shapefile)	ข้อมูลแผนที่ สำหรับการทำงานของ MapServer

3.5.3 โครงสร้างการทำงานของระบบ

โครงสร้างของระบบที่ใช้ในงานวิจัยจะพัฒนาโดยยึดโครงสร้างสถาปัตยกรรมระบบแบบ 4-Tiered GIS web service มาใช้เป็นต้นแบบ โดย ระบบที่พัฒนาจะประกอบด้วย 4 ส่วนหลักๆ โดยส่วนที่เป็น Web Server และ Map Server จะรวมการทำงานอยู่ที่เครื่องแม่ข่าย สำหรับรายละเอียดของส่วนประกอบต่างๆ มีดังต่อไปนี้

เว็บบราวเซอร์ (Web Browser): ใช้เป็นส่วนในการติดต่อผู้ใช้งานของระบบ

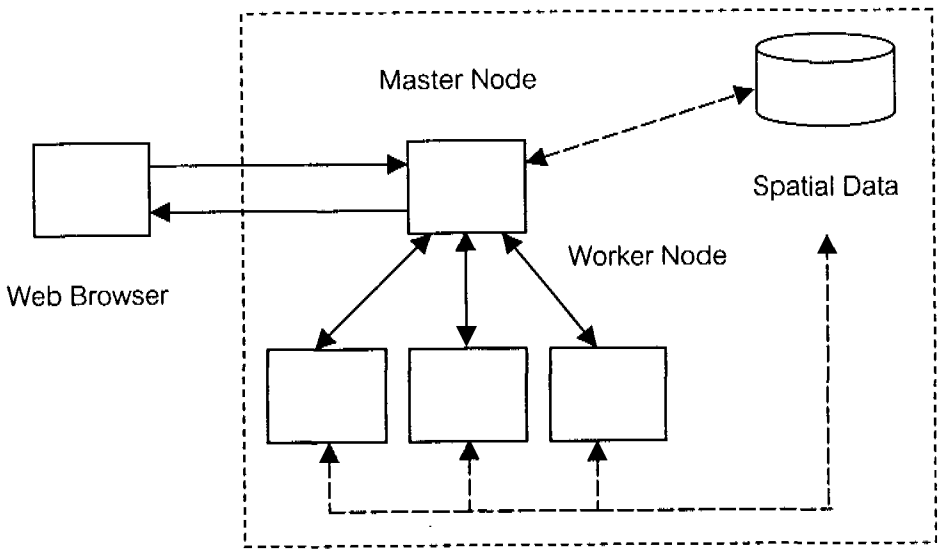
เครื่องแม่ข่าย (Server): ในส่วนเครื่องแม่ข่ายในงานวิจัยนี้เสนอการประยุกต์เครื่องแม่ข่ายโดยใช้ระบบพีซีคลัสเตอร์แบบแบวูฟล์เข้ามาทดแทน เพื่อเพิ่มประสิทธิภาพการประมวลผลให้กับระบบ โดยที่เครื่องแม่ข่ายที่เป็นแบวูฟล์คลัสเตอร์ จะประกอบด้วยเครื่องคอมพิวเตอร์ตั้งแต่ 2 เครื่องขึ้นไป (อาจจะเรียกว่า โหนด) มีโหนดหนึ่งรับทำหน้าที่เป็นโหนดหลัก (Master Node) ซึ่งจะเป็นโหนดที่ใช้ติดต่อสื่อสารกับระบบภายนอก นอกจากนั้นโหนดหลักจะใช้ในการรับ request จากผู้ใช้งาน และกระจายงานไปประมวลผลยังโหนดอื่นๆ รวมทั้งใช้เป็นโหนดที่รวบรวมผลลัพธ์การประมวลผลเพื่อส่งข้อมูลกลับไปยังผู้ใช้งานอีกด้วย ดังนั้นเมื่ออ้างอิงกับแนวคิดโครงสร้าง

สถาปัตยกรรมระบบแบบ 4-Tiered GIS web service เครื่องแม่ข่ายในงานวิจัยนี้จะทำหน้าที่เป็น Web Server และ Map Server ไปพร้อมๆ กัน

ข้อมูล (Data): สำหรับข้อมูลที่ระบบใช้ในการให้บริการ จะอยู่ในรูปแบบของข้อมูล GIS หรือข้อมูลเชิงพื้นที่ (Spatial Data) ซึ่งสามารถเทียบได้กับส่วนฐานข้อมูล (Database) ในโครงสร้างสถาปัตยกรรมระบบแบบ 4-Tiered GIS web service โดยข้อมูลเชิงพื้นที่จะถูกจัดสรรให้ทุกๆ โหนดในระบบสามารถเข้าถึงได้เท่าเทียมกันผ่านทางการแชร์ไฟล์ผ่านเครือข่าย Network-file system (NFS)

ภาพที่ 3.9

โครงสร้างของระบบโปรแกรม MapServer ที่พัฒนาขึ้นใหม่ (Parallel MapServer)

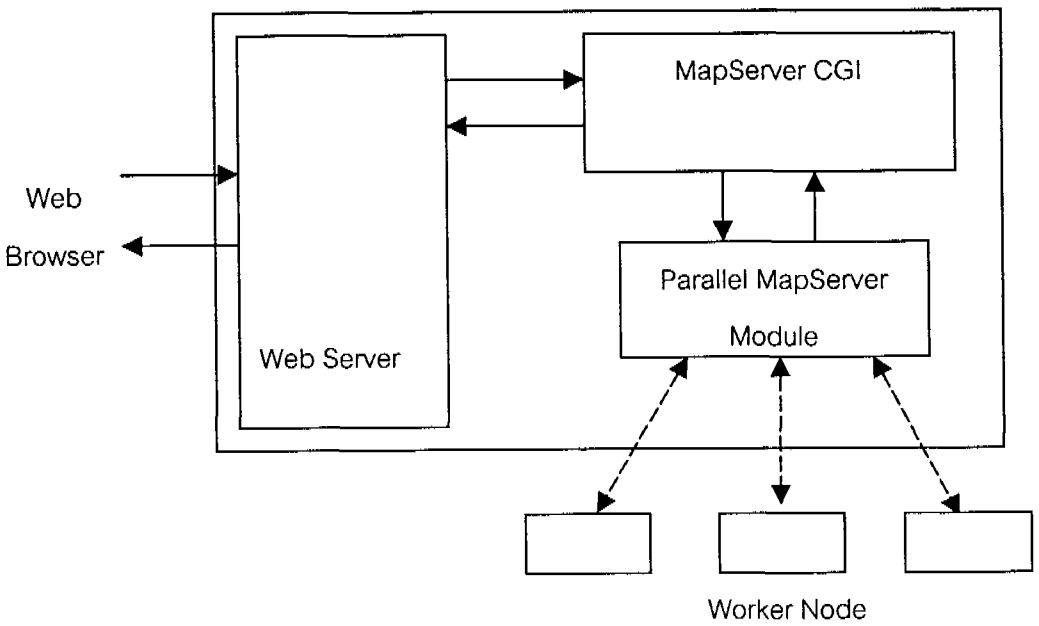


รายละเอียดโครงสร้างการทำงานของระบบ ระบบที่พัฒนาขึ้นในงานวิจัยนี้ จะถูกออกแบบให้แยกส่วนของระบบโปรแกรมที่พัฒนาใหม่ ซึ่งเป็นส่วนที่ประยุกต์ใช้แนวคิดการประมวลผลแบบขนาน ออกจากส่วนของโปรแกรมมาตรฐานดั้งเดิมของ MapServer โดยจะเรียกส่วนของโปรแกรมที่พัฒนาขึ้นใหม่นี้ว่า Parallel MapServer Module ลักษณะการทำงานโดยรวมของระบบโปรแกรม MapServer ที่พัฒนาขึ้นใหม่สามารถอธิบายได้ดังนี้:

เมื่อผู้ใช้งานส่ง Request เข้ามาผ่านทาง HTTP Protocol ระบบในส่วนที่เป็น Web Server จะรับคำร้องขอการทำงาน (Request) และจะส่งผ่าน Request ไปที่ MapServer CGI ผ่านทางการจัดเก็บค่าไว้ในตัวแปรของระบบ (Environment Variable) เมื่อ MapServer CGI รับ Request เข้ามาก็จะนำเอาคำสั่ง CGI command มาตรวจสอบว่าผู้ใช้งานต้องการทำงานกับฟังก์ชันใด ซึ่งถ้าตรวจสอบแล้วเป็นฟังก์ชันในส่วนที่ถูกพัฒนาขึ้นใหม่ในแนวคิดการประมวลผลแบบขนาน MapServer CGI จะทำการส่ง Request จากผู้ใช้งานพร้อมทั้งตัวแปรต่างๆ ที่จำเป็นในการประมวลผล ต่อมายังส่วนของโปรแกรมที่พัฒนาขึ้นใหม่ (Parallel MapServer Module) ซึ่งในที่นี้การส่ง Request ต่อมาของ MapServer CGI คือ การเรียกคำสั่งให้ทำการประมวลผลโปรแกรม Parallel MapServer Module แบบขนาน โดยเมื่อ MapServer CGI สั่งประมวลผล จะเกิดการสร้าง Process การทำงานขึ้นที่หน่วยประมวลผลหรือโหนดของระบบในแต่ละโหนด ทำการประมวลผลส่วนของโปรแกรมแบบขนานไปพร้อมๆ กัน และเมื่อประมวลผลเสร็จแต่ละโหนดจะส่งผลลัพธ์การประมวลผลกลับมาที่โหนดหลักเพื่อรวบรวมผลลัพธ์ของการประมวลผลทั้งหมดพร้อมทั้งส่งผลลัพธ์ที่ได้กลับไปที่ MapServer CGI เพื่อนำไปแสดงผลที่ Web Browser ของผู้ใช้งานต่อไป

ภาพที่ 3.10

โครงสร้างภายในระบบโปรแกรม MapServer ที่พัฒนาขึ้นใหม่



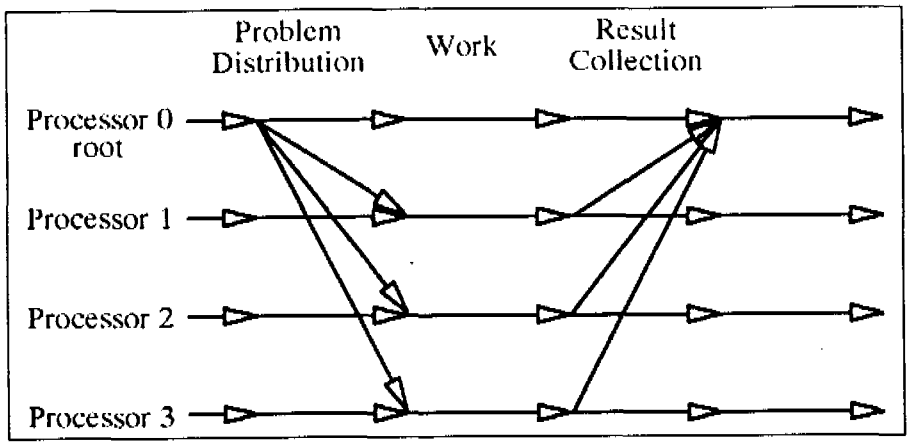
3.6 การออกแบบการกระจายงานและการประมวลผลแบบขนาน (Parallel and Distributed Design)

สืบเนื่องจากจุดมุ่งหมายหลักของการแก้ปัญหาในงานวิจัยนี้ ก็คือ การลดภาระในการประมวลผลข้อมูลอันเนื่องมาจากข้อมูลที่มีขนาดใหญ่ลง ดังนั้นการออกแบบระบบส่วนการกระจายงานและการประมวลผลแบบขนานในงานวิจัยนี้ จึงต้องการที่จะตัดแบ่งข้อมูลออกเป็นส่วนๆ ที่มีขนาดเล็ก แล้วให้แต่ละส่วนของข้อมูล ไปประมวลผลยังหน่วยประมวลผล (Processor) ที่แตกต่างกัน เพื่อให้หน่วยประมวลผล สามารถช่วยกันทำงานได้อย่างมีประสิทธิภาพที่สุด ซึ่งอาจจะกล่าวได้ว่าในงานวิจัยนี้ ออกแบบลักษณะการประมวลผลแบบขนานให้อยู่ในรูปแบบของการประมวลผลแบบขนานชนิด Data parallel นั่นคือ แต่ละโพรเซสที่เกิดขึ้นในขณะประมวลผลจะเกิดจากชุดคำสั่งเดียวกัน ทำงานในลักษณะเดียวกัน แต่จะประมวลผลหรือใช้ข้อมูลในส่วนที่แตกต่างกันออกไป

จากแนวทางการแก้ปัญหาข้างต้น งานวิจัยนี้จึงเลือกนำเอา แบบจำลองการส่งผ่านข้อความ (Message Passing Model) มาเป็นแบบจำลองหลักที่ใช้ในการออกแบบระบบ เหตุที่เลือกแบบจำลองดังกล่าวเนื่องจาก แบบจำลอง Message Passing มีลักษณะการทำงานที่สนับสนุนรูปแบบการประมวลผลแบบขนานแบบ Data parallel โดยในงานวิจัยจะทำการพัฒนาระบบอ้างอิงกับแบบจำลอง Message Passing โดยใช้เทคนิคการพัฒนาโปรแกรมแบบ SPMD (Single Program Multiple Data) ซึ่งในการประมวลผลงาน (Task) หรือโพรเซสแต่ละโพรเซสจะประมวลผลชุดคำสั่งเดียวกัน แต่จะประมวลผลกับข้อมูลที่แตกต่างกันไป และการสื่อสารส่งผ่านข้อมูลระหว่างโพรเซสจะใช้วิธีการส่งผ่านข้อความผ่านระบบเครือข่าย ซึ่งแนวทางการพัฒนานี้ ก็คือแนวคิดเดียวกับรูปแบบการประมวลผลแบบขนานชนิด Data parallel นั่นเอง

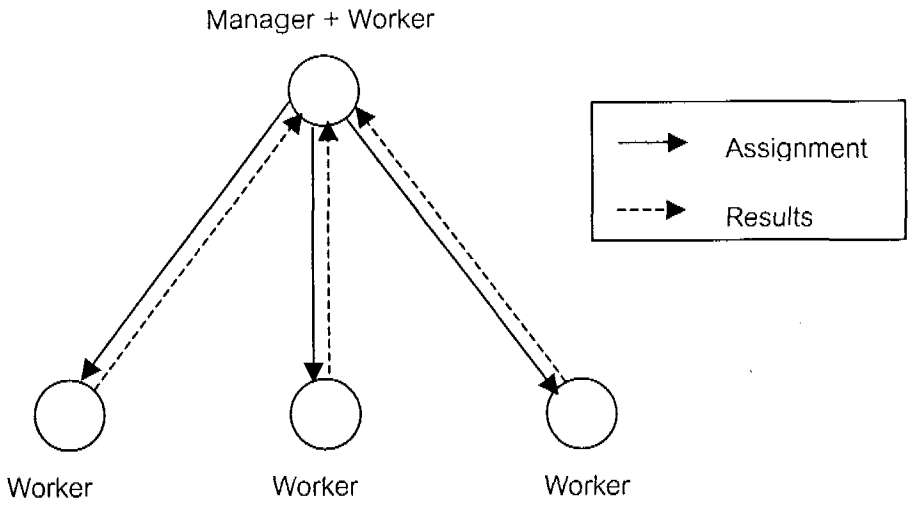
ภาพที่ 3.11

แสดงลักษณะการทำงานของแบบจำลองการส่งผ่านข้อความ (Message Passing Model)



สำหรับในส่วนของรูปแบบการกระจายงานหรืออัลกอริทึมสำหรับการจัดการโพรเซสที่เกิดขึ้นระหว่างการประมวลผลแบบขนานนั้น ในงานวิจัยนี้ใช้การกระจายงานที่มีรูปแบบคล้ายกับการกระจายงานแบบ Manager-Worker ลักษณะการทำงาน โพรเซสของ CGI เป็นจุดเริ่มต้นในการสั่งให้แยกประมวลผลแบบขนาน ที่จุดนี้จะเกิดการสร้างโพรเซสขึ้นเพื่อทำการคำนวณที่แต่ละหน่วยประมวลผล จากนั้นหลังจากที่โพรเซสที่ทำงานบนแต่ละหน่วยประมวลผลทำการคำนวณเสร็จเรียบร้อยจะส่งผลลัพธ์กลับมาที่โพรเซสหลักเพื่อรวบรวมและส่งผลลัพธ์กลับไปที่โพรเซสของ CGI อีกครั้งก่อนนำไปแสดงผลที่ฝั่งผู้ใช้งาน

ภาพที่ 3.12
แผนภาพจำลองรูปแบบการกระจายงานที่ใช้ในงานวิจัย



การทำงานระบบจะกระจายงานไปประมวลผลยังหน่วยประมวลผลต่างๆ ที่อยู่ในระบบ โดยการกระจายงานมีจุดเริ่มต้นที่โปรแกรม CGI เมื่อผู้ใช้งานส่งคำร้องขอการทำงาน (Request) เข้ามาจะเกิดการสร้างโพรเซสของ MapServer CGI ขึ้นเพื่อรับ Reqeust ของผู้ใช้งาน จากนั้น CGI จะกระจายงานโดยเรียกคำสั่งให้เกิดการประมวลผลแบบขนานพร้อมทั้งระบุจำนวนโพรเซสที่ต้องการสร้างไปยัง Parallel MapServer Module ตามโครงสร้างของระบบที่กล่าวไว้ในหัวข้อโครงสร้างระบบข้างต้น ระบบจะทำการสร้างโพรเซสขึ้นทั้งที่โหนดหลัก (Master node) และโหนดที่ใช้ในการทำงาน (Worker node) และประมวลผลโพรเซสที่แต่ละหน่วยประมวลผล โดยใช้คำสั่งการกำหนดเงื่อนไข (Conditional Branch) if-then-else เป็นคำสั่งที่ใช้ในการเลือกชุดคำสั่งและส่วนของข้อมูลที่เหมาะสมกับโพรเซสในแต่ละหน่วยประมวลผล ในการประมวลผลแต่ละโพรเซสจะประมวลผลเฉพาะส่วนของข้อมูลที่ตนเองรับผิดชอบ จากนั้นเมื่อประมวลผลเสร็จ แต่ละโพรเซสจะส่งผลลัพธ์ที่ได้กลับมารวบรวมที่โพรเซสหลักที่สร้างขึ้นบนโหนดหลัก แล้วจึงส่งผลลัพธ์กลับไปที่ MapServer CGI เพื่อเตรียมการแสดงผลข้อมูลต่อผู้ใช้งานอีกครั้งหนึ่ง

จากรายละเอียดรูปแบบการกระจายงานแสดงให้เห็นว่า โพรเซสของ MapServer CGI จะทำหน้าที่คล้ายกับเป็นโพรเซส Manager ที่คอยจัดการสำหรับการกระจายงานและสั่งให้เกิดการสร้างโพรเซสสำหรับการทำงานแบบขนาน อีกทั้งยังเป็นโพรเซสกลางที่ทำการรวบรวมผลลัพธ์ที่ได้จากการประมวลผล ก่อนนำผลลัพธ์ที่ได้ไปแสดงผลอีกด้วย ด้วยเหตุนี้จึงทำให้แนวทางการ

กระจายงานที่ใช้ในงานวิจัยนี้ มีรูปแบบคล้ายกับการกระจายงานในรูปแบบ Manager-Worker เป็นอย่างยิ่ง

3.7 แนวทางการวิเคราะห์ผลการทดสอบ

การวิเคราะห์ผลการทดสอบ เป็นกระบวนการที่นำเอาผลลัพธ์ที่ได้จากการทดสอบระบบโปรแกรม MapServer ที่ถูกพัฒนาขึ้นมาใหม่ กับระบบโปรแกรม MapServer แบบดั้งเดิมมาทำการวิเคราะห์เพื่อหาผลสรุปที่ได้จากการทดสอบระบบ โดยที่ในงานวิจัยนี้จะทำการวิเคราะห์ผลลัพธ์ของการทดสอบ ซึ่งก็คือ เวลาที่ระบบประมวลผล (elapse times) อ้างอิงกับตัวชี้วัด 2 ตัว คือ ความเร็วของระบบใหม่ที่เพิ่มขึ้นเมื่อเทียบกับระบบเดิม (Speedup) และ ประสิทธิภาพในการใช้งานหน่วยประมวลผล (Processor Efficiency) ผลสรุปที่ได้จากการเปรียบเทียบตัวชี้วัด จะแสดงให้เห็นถึงความสำเร็จของการเพิ่มประสิทธิภาพของระบบโปรแกรม Web-Based GIS ที่นำเสนอในงานวิจัย

ความเร็วของระบบใหม่ที่เพิ่มขึ้นเมื่อเทียบกับระบบเดิม (Speedup) สามารถหาได้จากเวลาการประมวลผลทั้งหมดที่ 1 หน่วยประมวลผล (processor) หรือ เวลาการประมวลผลของโปรแกรมแบบลำดับ (t_1) เทียบกับ เวลาการประมวลผลที่หลายหน่วยประมวลผล หรือ เวลาการประมวลผลของโปรแกรมแบบขนาน (t_n) โดยที่ n คือ จำนวนหน่วยประมวลผลแสดงการคำนวณความเร็วที่เพิ่มขึ้น (Speedup) ดังสมการที่ 3.1

$$\text{Speedup}_n = \frac{t_1}{t_n} \tag{3.1}$$

โดยนิยามแล้ว เมื่อทำการเพิ่มจำนวนหน่วยประมวลผลมากขึ้น ความเร็วในการประมวลผลของโปรแกรมก็จะเพิ่มขึ้นตามจำนวนหน่วยประมวลผลด้วย และถ้าประสิทธิภาพในการใช้งานหน่วยประมวลผลมีประสิทธิภาพสูงสุดในทางอุดมคติ คือ ใช้ประโยชน์หน่วยประมวลผลได้ประสิทธิภาพ 100% แล้ว จะก่อให้เกิดความเร็วที่เพิ่มขึ้น (Speedup) ในทางอุดมคติที่เรียกว่า linear speedup

แต่ในความเป็นจริง โอกาสที่จะเกิด linear speedup เป็นไปได้ยาก ทั้งนี้เนื่องมาจากในการทำงานของโปรแกรมในสถานการณ์จริง มีตัวแปรหลายตัวแปรที่เป็นข้อจำกัดทำให้ความเร็วที่เพิ่มขึ้น (speedup) นั้นน้อยกว่า linear speedup ตัวแปรที่เป็นข้อจำกัดเหล่านั้นได้แก่ เวลาที่เสียไปในการติดต่อสื่อสารระหว่างโพรเซสที่เกิดขึ้นในอัลกอริทึมแบบขนาน (Communication-Overhead) การ load balancing งานระหว่างโหนดที่ไม่ดีพอ รวมไปถึงเวลาที่เสียไปจากส่วนของอัลกอรึทึมต่างๆ ในโปรแกรมที่พัฒนาขึ้นมาไม่มีประสิทธิภาพดีเพียงพอ เป็นต้น

ส่วนประสิทธิภาพในการใช้งานหน่วยประมวลผล (Processor Efficiency) สามารถคำนวณได้จาก ความเร็วของระบบใหม่ที่เพิ่มขึ้นเมื่อเทียบกับระบบเดิม เทียบกับจำนวนหน่วยประมวลผล แล้วทำให้เป็นร้อยละโดยคูณด้วยหนึ่งร้อย แสดงการคำนวณประสิทธิภาพการใช้งานหน่วยประมวลผลดังสมการที่ 3.2

$$\text{efficiency} = \frac{\text{speedup} * 100}{\text{number of processor}} \qquad (3.2)$$

สำหรับการแสดงผลลัพธ์ของการวิเคราะห์ ในงานวิจัยนี้จะนำเสนอผลของการวิเคราะห์เปรียบเทียบตัวชี้วัดโดยใช้แผนภูมิเส้นแสดงแนวโน้มการเปลี่ยนแปลงค่าของตัวชี้วัด เมื่อจำนวนหน่วยประมวลผลเพิ่มมากขึ้นเป็นค่าทวีคูณ เหตุผลที่แสดงผลการวิเคราะห์โดยใช้แผนภูมิเพื่อที่จะสามารถแสดงผลลัพธ์ของการทดลองรวมทั้งการวิเคราะห์ได้อย่างชัดเจน สามารถเปรียบเทียบหรือแสดงผลการทดสอบได้อย่างเป็นรูปธรรมและสามารถเข้าใจได้ง่าย