ระบบซูเปอร์คอมพิวเตอร์ราคาประหยัด

POOR'S MAN SUPERCOMPUTER

นายมงคล พิทักษ์สุขสันติ

นายสาโรจน์ พงษ์พานิช

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ระบบซูเปอร์คอมพิวเตอร์ราคาประหยัด

POOR'S MAN SUPERCOMPUTER

โดย

นายมงคล พิทักษ์สุขสันติ

นายสาโรจน์ พงษ์พานิช

อาจารย์ที่ปรึกษา

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบซูเปอร์คอมพิวเตอร์ราคาประหยัด

POOR'S MAN SUPERCOMPUTER

ผู้จัดทำ

1. นายมงคล พิทักษ์สุขสันติ รหัสประจำตัว 40010576
2. นายสาโรจน์ พงษ์พานิช รหัสประจำตัว 40010844

อาจารย์ที่ปรึกษา

(อาจารย์อัครเดช วัชระภูพงษ์)

ระบบซูเปอร์คอมพิวเตอร์ราคาประหยัด

นายมงคล พิทักษ์สุขสันติ 40010576
นายสาโรจน์ พงษ์พานิช 40010844
อาจารย์อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ในโลกยุคปัจจุบันการใช้งานคอมพิวเตอร์มีความสำคัญต่อชีวิตประจำวันของมนุษย์เป็นอย่างมาก คอมพิวเตอร์ที่มีประสิทธิภาพสูงจึงมีความจำเป็นต่อการใช้งาน เพื่อให้ทันต่อความต้องการ จึงมีการผลิตเครื่องคอมพิวเตอร์เพื่อตอบสนองความต้องการดังกล่าวมากมายหลายค่าย โดยทั่วไปจะเรียกเครื่องเหล่านี้ว่า " ซูเปอร์คอมพิวเตอร์ " เครื่องคอมพิวเตอร์เหล่านี้มีความสามารถในการคิดคำนวณได้อย่างรวดเร็วโดยใช้เทคโนโลยีระดับสูง แต่จากการที่ต้องใช้เทคโนโลยีระดับสูง จึงทำให้มีราคาของผลิตภัณฑ์ค่อนข้างสูง จึงได้มีการคิดค้นระบบคอมพิวเตอร์ที่สามารถทำงานได้อย่างมีประสิทธิภาพทดแทนเครื่องซูเปอร์คอมพิวเตอร์เหล่านั้นในราคาที่พอเหมาะ ระบบดังกล่าวมานี้คือ " ดิสก์เลส คลัสเตอร์ " ซึ่งใช้เทคโนโลยีคลัสเตอร์ในการทำงาน

วิทยานิพนธ์ฉบับนี้เป็นแนวทางในทดลองสร้างระบบดิสก์เลสคลัสเตอร์ ซึ่งประกอบด้วยไลบรารีใช้ในการประมวลผลแบบขนาน และโปรแกรมที่ช่วยเพิ่มประสิทธิภาพในการทำงานของระบบดิสก์เลสคลัสเตอร์ พร้อมทั้งได้ทำการทดสอบประสิทธิภาพการทำงานของระบบด้วย และจากการทดสอบ สรุปได้ว่าการทำระบบดิสก์เลสคลัสเตอร์นั้นขึ้นองค์ประกอบหลายๆอย่าง ทั้งทางด้านซอฟท์แวร์และ ฮาร์ดแวร์ โดยจะต้องมีส่วนประกอบทั้งสองที่พอเหมาะ จึงจะทำให้ได้ประสิทธิภาพที่ดี

POOR'S MAN SUPERCOMPUTER

Mr.Mongkol Phitaksuksanti

Mr.Sarote Pongpanich

Mr.Akkradach Watcharapupong Advisor

ABSTRACT

Today computers play a very important role in our daily life. Therefore, a computer with high performance is needed. Many vendors have tried to design those computers in order that they can be used in every aspect. Generally, they are called "SuperComputer" - these computers are capable of computing quickly by utilizing high technology. However, due to the use of this high technology, the price goes up. To reduce the cost, an effective computer system is invented to replace "SuperComputer" with a reasonable price. This above-mentioned system is called "Diskless Cluster".

This thesis is concerned with diskless cluster that consists of library and program that use for parallel processing in diskless cluster and test it. Conclusion is diskless cluster depend on multiple component - both hardware and software. However the system should have suitable component for the best performance.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลาย ๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจเอาใจใส่เสมอมา ในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

มงคล พิทักษ์สุขสันติ

สาโรจน์ พงษ์พานิช

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย I
บทคัดย่อภาษาอังกฤษ II
กิตติกรรมประกาศ III
สารบัญ IV
สารบัญตาราง VII
สารบัญภาพ VIII
บทที่ 1 บทนำ 1
1.1 ความสำคัญและความเป็นมา 1
1.2 ขอบเขตของงานวิจัย 2
1.3 ประโยชน์ที่คาดว่าจะได้รับ 2
1.4 การดำเนินงาน 2
บทที่ 2 ซูเปอร์คอมพิวเตอร์ 3
2.1 บทนำ 3
2.2 รูปแบบที่ใช้ภายในระบบซูเปอร์คอมพิวเตอร์ 3
2.2.1 มัลติโพรเซสซิง 3
2.2.2 มัลติฟังก์ชันโพรเซสเซอร์ 5
2.2.3 ไปป์ไลน์โพรเซสเซอร์ 6
2.2.4 อาร์เรย์โพรเซสเซอร์ 7
2.3 การประมวลผลแบบขนาน 8
2.3.1 มัลติโพรเซสซิง 8
2.3.2 โพรเซสเซอร์แบบขนาน 9
2.3.3 การคำนวณแบบเวคเตอร์ 10
2.3.4 การคำนวณทางเวคเตอร์ 10
2.3.5 เชนนิ่ง 14
2.4 สถาปัตยกรรมของระบบคอมพิวเตอร์ 14
2.4.1 Lookahead 16
2.4.2 Explicit vector instructions 16
2.4.3 Flynn classification 16
2.4.4 คอมพิวเตอร์แบบขนาน และ เวคเตอร์คอมพิวเตอร์ 19

สารบัญ(ต่อ)

หน้าที่

บทที่ 3 Beowulf 20

3.1 บทนำ 20

3.2 โครงสร้างและประสิทธิภาพในการทำงาน 20

3.3 เทคโนโลยีทางด้านเครือข่ายและการขยายเครือข่าย 21

3.4 ซอฟต์แวร์ที่ใช้ในระบบรวมถึงโปรแกรมที่ช่วยในการทำงาน 22

3.5 การขยายขนาดของระบบ 24

3.6 การเกิดคอขวดทางฮาร์ดแวร์ 25

3.7 การเกิดคอขวดทางซอฟต์แวร์ 26

3.8 การเปรียบเทียบระหว่าง Symmetric Multiprocessor กับ คลัสเตอร์ 27

3.8.1 หัวข้อในการเปรียบเทียบ 27

3.8.2 การพิจารณาในรายละเอียด 28

3.9 เทคโนโลยีการเชื่อมต่อแบบและการพัฒนาในรูปแบบต่างๆ 29

บทที่ 4 ระบบคลัสเตอร์ที่ได้ทำการออกแบบ 31

4.1 บทนำ 31

4.2 การทำงานของดิสก์เลส 32

4.3 การทำงานของ NIS 34

4.4 การทำงานของ NFS 34

4.5 โครงสร้างการทำงานของ PVM 35

4.5.1 ภาพรวมของ PVM 35

4.5.2 ระบบของ PVM 35

4.5.3 ขั้นตอนการติดตั้ง PVM 40

4.5.4 การเริ่มใช้งานและหยุดการใช้งาน PVM 40

4.5.5 คำสั่งที่ใช้งานบนคอนโซลของ PVM 41

4.5.6 Host file Options 42

4.5.7 จุดประสงค์หลักของ PVM3 43

4.6 โครงสร้างการทำงานของ MPI 44

4.6.1 ความรู้เบื้องต้นเกี่ยวกับ MPI 44

4.6.2 จุดมุ่งหมายของ MPI 45

4.6.3 ภาษาที่สามารถนำมาใช้ร่วมกับ MPI ได้ 46

สารบัญ(ต่อ)

หน้าที่

4.7 โปรแกรมมอนิเตอร์ 46
4.8 โปรแกรมโมสิคซ์ 49
บทที่ 5 การใช้งานและทดสอบระบบ 51
บทที่ 6 วิจารณ์และสรุปผลจากการทำงานของระบบ 54
บรรณานุกรม 56

สารบัญตาราง

หน้าที่

ตารางที่ 2.1 แสดงการเปรียบเทียบรูปแบบการทำงานในระบบซูเปอร์คอมพิวเตอร์แบบต่าง ๆ 8
ตารางที่ 3.1 แสดง tools ต่างที่กำลังอยู่ระหว่างการพัฒนา 23
ตารางที่ 4.1 แสดงสถาปัตยกรรมที่สนับสนุน PVM (PVM_ARCH names used in PVM 3) 39
ตารางที่ 5.1 แสดงผลการทดสอบระบบด้วยโปรแกรม bladeenc โดยผ่าน MPI 52

สารบัญภาพ

หน้าที่

รูปที่ 2.1 แสดงซูเปอร์คอมพิวเตอร์ที่ใช้ Shared - memory 4
รูปที่ 2.2 แสดงซูเปอร์คอมพิวเตอร์ที่ใช้ Distributed – memory 4
รูปที่ 2.3 แสดงรูปแบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน 5
รูปที่ 2.4 แสดงตัวอย่างของ functional units 6
รูปที่ 2.5 แสดง Array processor 7
รูปที่ 2.6 แสดงการแบ่งชนิดของ Parallel Processors 9
รูปที่ 2.7 แสดงตัวอย่างของการบวกเวคเตอร์ 10
รูปที่ 2.8 แสดง Scalar Processing 11
รูปที่ 2.9 แสดง Scalar Processing 11
รูปที่ 2.10 แสดง Parallel Processing 11
รูปที่ 2.11 แสดง Pipelined ALU 12
รูปที่ 2.12 แสดง Parallel ALUs. 12
รูปที่ 2.13 แสดงการประมวลผลแบบไปป์ไลน์ 13
รูปที่ 2.14 แสดงวิวัฒนาการของสถาปัตยกรรมคอมพิวเตอร์ 15
รูปที่ 2.15 แสดง SISD uniprocessor architecture 16
รูปที่ 2.16 แสดง SIMD architecture (with distributed memory) 17
รูปที่ 2.17 แสดง MIMD architecture (with shared memory) 17
รูปที่ 2.18 แสดง MIMD architecture (with distributed memory) 18
รูปที่ 2.19 แสดง MISD architecture (the systolic array) 18
รูปที่ 3.1 แสดงการเปรียบเทียบประสิทธิภาพเมื่อจำนวนโปรเซสเซอร์มีจำนวนเพิ่มขึ้น 28
รูปที่ 3.2 วิวัฒนาการของ Networking speed 29
รูปที่ 4.1 ภาพรวมของระบบ PVM (PVM System Overview) 37
รูปที่ 4.2 การเชื่อมต่อของสถาปัตยกรรมต่างๆในระบบ PVM 38
รูปที่ 4.3 หน้าจอแสดงผลแบบกราฟิกของ XPVM 40
รูปที่ 4.4 แสดงการเขียน hostfile ที่ใช้คำสั่งต่างๆ 43
รูปที่ 4.5 แสดงเครื่องทั้งหมดที่อยู่ในคลัสเตอร์ 47
รูปที่ 4.6 แสดงสถานะว่ามีเครื่องในในคลัสเตอร์ที่ทำงานอยู่ในปัจจุบัน 48
รูปที่ 4.7 แสดงการใช้งานดิสก์ในแต่ละเครื่องโหนด 48
รูปที่ 4.8 แสดงการค่าติดตั้งในแต่ละเครื่องโหนด 49

สารบัญภาพ(ต่อ)

หน้าที่

รูปที่ 5.1 กราฟแสดงการเปรียบเทียบ ประสิทธิภาพของการทดสอบระบบ แบบ MPI 53

บทที่ 1
บทนำ

1.1 ความสำคัญและความเป็นมา

ในปัจจุบันเทคโนโลยีต่างๆ ก้าวไปอย่างรวดเร็ว มีความก้าวหน้าทางเทคโนโลยีอย่างต่อเนื่อง เช่นเดียวกับศาสตร์ทางด้านคอมพิวเตอร์ที่เติบโตอย่างรวดเร็วในช่วงไม่กี่ปีที่ผ่านมา ในทุกวันนี้องค์กรที่มีการคำนวณทางวิทยาศาสตร์ หรือ การคำนวณคณิตศาสตร์ระดับสูง แทบจะไม่มีที่ใดเลยที่จะไม่มีเครื่องคอมพิวเตอร์มาช่วยในงานเช่นนี้ เหตุที่เป็นเช่นนี้ก็เพราะเครื่องคอมพิวเตอร์เป็นอุปกรณ์ที่มีความสามารถในการคำนวณทางคณิตศาสตร์ได้อย่างรวดเร็ว แต่เนื่องจากความเจริญก้าวหน้า และการใช้งานทางวิทยาศาสตร์ และคณิตศาสตร์ มีอยู่ทุกวัน และมีแนวโน้มที่จะเพิ่มขึ้น การเพิ่มขึ้นไม่เฉพาะจำนวนงาน แต่รวมถึงขนาดงาน และรวมถึงจำนวนองค์กรที่ใช้ ทำให้เครื่องคอมพิวเตอร์(พีซี) ที่ใช้อยู่นั้นมีความสามารถด้อยเกินไป หากต้องนำมาคิดคำนวณทางคณิตศาสตร์ระดับสูง จึงมีการคิดค้นสร้างระบบที่เรียกว่า ซูเปอร์คอมพิวเตอร์ (SuperComputer)

ในปัจจุบันมีซูเปอร์คอมพิวเตอร์ออกมาขายหลายค่ายด้วยกัน ซึ่งแต่ละค่ายก็มีความโดดเด่นเป็นของตัวเอง และส่วนใหญ่ต้องเขียนโปรแกรมที่สามารถทำงานได้เฉพาะเครื่องในรุ่นๆนั้น เท่านั้น ทำให้เกิดปัญหาเมื่อต้องการนำโปรแกรมที่เขียนขึ้นไปทำงานที่อื่น และอีกปัญหาที่สำคัญมากสำหรับโลกยุคปัจจุบันนี้ที่มีการขยายตัวอยู่ตลอดเวลา มีการแข่งขันทั้งทางด้านการศึกษา รวมถึงทางด้านธุรกิจ โดยเฉพาะทางด้านธุรกิจมักต้องการความรวดเร็วฉับพลันในการทำงานค่อนข้างมาก จึงต้องมีการซื้ออุปกรณ์ที่เรียกว่า "ซูเปอร์คอมพิวเตอร์" ที่ได้กล่าวถึงมาแล้วก่อนหน้านี้ แต่ปัญหาสำคัญที่เกิดขึ้นเสมอกับเครื่องซูเปอร์คอมพิวเตอร์ คือ ความสามารถในการขยายระบบของซูเปอร์คอมพิวเตอร์นั้นเป็นไปได้อย่างจำกัดและมีขีดสุดของความสามารถของเครื่องนั้น จึงได้มีการคิดค้นและสร้างระบบที่มีจุดประสงค์ คือ สามารถทำงานได้ในแนวทาง ๆ เดียวกับซูเปอร์คอมพิวเตอร์ จึงได้มีระบบที่เรียกว่า Beowulf เกิดขึ้นเพื่อ สนองความต้องนี้ โดยระบบ Beowulf ที่มีการออกแบบมานั้น มีการออกแบบและสร้างบนปัจจัย 3 ข้อหลักคือ " ดีกว่าเดิม เร็วกว่าเดิม และถูกกว่าเดิม" โดยคำว่า "ดีกว่าเดิม" จะหมายถึง สามารถสร้างระบบที่มีความสามารถที่จะทำงานในส่วนที่ระบบซูเปอร์คอมพิวเตอร์เดิมทำไม่ได้ คำว่า "เร็วกว่าเดิม" หมายถึง สามารถสร้างระบบที่สามารถปรับแต่งระบบให้มีความเร็วที่สูงกว่าเดิม และ คำว่า "ถูกกว่าเดิม" การพัฒนาดังกล่าวถึงแม้ปัจจุบันยังไม่สามารถทำได้เสมือนอยู่บนซูเปอร์คอมพิวเตอร์ แต่ก็สามารถทำให้มีประสิทธิภาพไม่ด้อยกว่าเท่าไรนัก และที่สำคัญระบบคลัสเตอร์สามารถขยายขนาดของระบบอย่างแทบไม่มีขีดจำกัด โดยอาจจะมีบ้างก็คือ ส่วนของระบบเครือข่ายที่อาจไม่สามารถสนับสนุนการทำงานของระบบขนาดใหญ่ได้ แต่ก็สามารถทดแทนด้วยระบบเครือข่ายที่มีประสิทธิภาพสูงขึ้น แต่ก็อาจมีราคาสูงขึ้นตามความสามารถ

1.2 ขอบเขตของงานวิจัย

ขอบเขตของงานที่ทำ ส่วนแรกคือ การศึกษาระบบอย่างคร่าว ของระบบคลัสเตอร์ที่ใช้งานปัจจุบัน ส่วนสองคือ หลังจากที่ได้ศึกษามาแล้วก็ต้องพยายามออกแบบระบบให้มีประสิทธิภาพ แต่ต้องทำการตัดทรัพยากรส่วนที่ไม่จำเป็นออก โดยระบบทั้งระบบต้องใช้ทรัพยากรที่เหลือใช้ และส่วนสุดท้าย คือ สร้างระบบตัวอย่างที่สามารถทำงานได้จริง พยายามออกแบบตามแนวคิด เพื่อระบบให้มีความสามารถในการทำงานได้อย่างมีประสิทธิภาพตามแนวทางของหัวข้อโครงงาน

โดยเมื่อศึกษา, ออกแบบ, และสร้างแล้ว จะต้องตรวจสอบประสิทธิภาพที่ได้รับ เมื่อใช้งานจริง และสามารถนำข้อมูลที่ได้มาเปรียบเทียบ และวิเคราะห์ถึงรายละเอียดได้

1.3 ประโยชน์ที่คาดว่าจะได้รับ

- ได้เข้าใจถึงระบบ โครงสร้างและรูปแบบการทำงานของระบบพีซีความเร็วสูง
- นำความรู้ที่ได้นำมาสร้างระบบคอมพิวเตอร์ประมวลผลเชิงวิทยาศาสตร์และวิศวกรรมศาสตร์ความเร็วสูงสำหรับงานวิจัยต่าง ๆ
- เพื่อนำไปประยุกต์ใช้ออกแบบและ สร้างระบบพีซีความเร็วสูงเพื่อใช้งานจริง

1.4 วิธีการดำเนินงาน

งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่างๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งก็มีเรื่องหลัก ๆ อยู่ 2 เรื่องด้วยกัน คือ ระบบซูเปอร์คอมพิวเตอร์ และระบบคลัสเตอร์ซึ่งในที่นี้ได้ศึกษาระบบ Beowulf ซึ่งมีรายละเอียดดังในบทที่ 2 และ 3

จากนั้นก็จะเริ่มเข้าสู่ขั้นตอนของการพัฒนาระบบ โดยบทที่ 4 จะกล่าวถึงองค์ประกอบโดยรวมของระบบที่พัฒนาขึ้นมาทั้งหมด และยังอธิบายไปถึงการทำงานของโปรแกรมต่างๆ ที่ได้พัฒนาลงในระบบ

สำหรับบทที่ 5 ก็จะแสดงวิธีการใช้งานระบบและผลการทดสอบระบบรวมทั้งหมด และบทที่ 6 ซึ่งเป็นบทสุดท้ายก็จะเป็นการวิจารณ์และสรุปผลการทำงานของระบบ และแนวทางในการพัฒนางานวิจัยนี้เพิ่มเติม และแนวทางในการนำไปประยุกต์ใช้

บทที่ 2
ซูเปอร์คอมพิวเตอร์

2.1 บทนำ

ซูเปอร์คอมพิวเตอร์ คือ คอมพิวเตอร์ที่มีอัตราการคำนวณสูงที่สุด หน่วยความจำใหญ่ที่สุด มีขนาดใหญ่ที่สุดและราคาสูงสุด บางครั้งอาจเรียกได้ว่า " เครื่องบดตัวเลข " นั่นคือ เครื่องจักรที่ถูกออกแบบมาให้แสดงผลการคำนวณทางตัวเลขด้วยความเร็วสูงสุด โดยการใช้เทคโนโลยีสูงสุดในการออกแบบ บ่อยครั้งที่เครื่องซูเปอร์คอมพิวเตอร์ได้รับการนำมาใช้แก้ปัญหาด้านต่าง ๆ เช่น การแก้ปัญหาทางตัวเลขที่เกี่ยวกับปรากฏการณ์ต่าง ๆ ทางด้านฟิสิกส์, การพยากรณ์อากาศ, การวิเคราะห์ทางด้านวัสดุศาสตร์, ฟิสิกส์ด้านพลังงาน, การสำรวจน้ำมัน, การออกแบบวงจรไฟฟ้า และการออกแบบเครื่องบิน อย่างไรก็ตาม แม้ว่าเครื่องซูเปอร์คอมพิวเตอร์จะมีความสามารถในการประมวลผลคำสั่งถึงหลายร้อยล้านคำสั่งต่อวินาที แต่ก็ยังมีข้อจำกัดในเรื่องของความเที่ยงตรงอยู่ และปัญหาที่ใหญ่ที่สุดของเครื่องซูเปอร์คอมพิวเตอร์ก็คือ ราคา เนื่องจากเครื่องซูเปอร์คอมพิวเตอร์ใช้เทคโนโลยีระดับสูงในการออกแบบและทำงานนั่นเอง อีกทั้งเครื่องซูเปอร์คอมพิวเตอร์ยังต้องการโปรแกรมที่สามารถรับประกันได้ว่าจะสนับสนุนการทำงานของเครื่องได้อย่างสมบูรณ์อีกด้วย

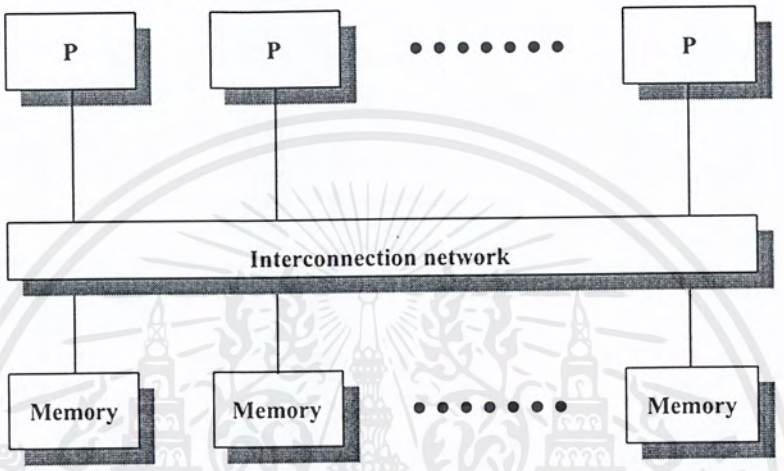
2.2 รูปแบบที่ใช้ภายในระบบซูเปอร์คอมพิวเตอร์

2.2.1 มัลติโพรเซสซิง (Multiprocessing) แต่ละโพรเซสเซอร์ในเครื่องคอมพิวเตอร์ที่ทำงานแบบมัลติโพรเซสซิงนี้ สามารถทำการเอ็กซคิวคำสั่งได้โดยไม่ขึ้นกับส่วนอื่น ๆ โดยคุณสมบัติแบบขนานได้มาจากการจำแนกโปรแกรมออกเป็นส่วนย่อย ๆ และโพรเซสเซอร์แต่ละตัวสามารถเอ็กซคิวโปรแกรมได้พร้อมกัน เราสามารถจำแนกระบบมัลติโพรเซสซิงได้เป็น 2 แบบ ดังนี้

- **Shared - memory** ในคอมพิวเตอร์ชนิดนี้ ไมโครโพรเซสเซอร์จำนวน 8 – 128 ตัว หรือโพรเซสเซอร์ประสิทธิภาพสูงหลาย ๆ ตัว จะเชื่อมต่อเข้ากับชุดของหน่วยความจำ (Memory module) โดยทำงานผ่านทางระบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน (Interconnection network) ชุดของหน่วยความจำนี้สามารถเข้าถึงได้โดยโพรเซสเซอร์ใด ๆ และการอ้างถึงหน่วยความจำหลัก (แอดเดรสสเปซ) โปรแกรมที่เอ็กซคิวแล้ว, ข้อมูล และผลลัพธ์จะเก็บไว้ในส่วนของหน่วยความจำที่ได้รับการอ้างถึง ซึ่งรูปแบบนี้เรียกได้อีกอย่างหนึ่งว่า " tightly coupled multiprocessor system "

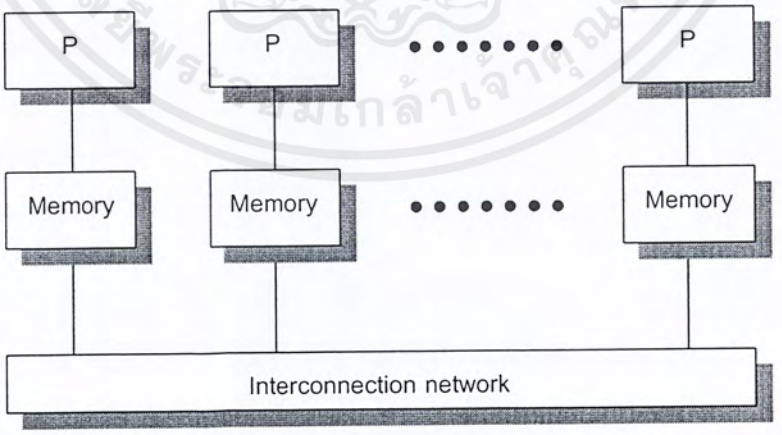
ระบบการเชื่อมต่อเน็ตเวิร์กระหว่างกันของแบบ shared - memory นี้ ส่วนมากเป็นแบบบัส (bus) ดังรูปที่ 2.1 ซึ่งระบบแบบบัสนี้บัสจะแบ่งกันใช้โดยโพรเซสเซอร์แต่ละตัวและหน่วยความจำ โดยที่สองโพรเซสเซอร์ไม่สามารถใช้บัสในเวลาเดียวกันได้ จึงทำให้ต้องมีการจัดคิว (queue) ขึ้นเพื่อให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

การทำงานร่วมกันระหว่างโพรเซสเซอร์ใช้วิธีการที่เรียกว่า Execute Synchronization Instructions ที่จะทำงานบน shared - memory modules หรือที่อุปกรณ์พิเศษบางตัว shared - memory multiprocessor เกิดขึ้นตั้งแต่ปี ค.ศ. 1960 โดยเครื่อง Burroughs B500 และ D – 825 อย่างไรก็ตามการประมวลผลแบบมัลติโพรเซสซิงใช้เพื่อเพิ่มความเชื่อถือได้ของระบบและเพื่อให้สามารถประมวลผลและจัดการอินพุทและเอาท์พุทได้พร้อมกัน ตั้งแต่การเปิดตัวของเครื่องคอมพิวเตอร์ Cray X – MP ในปี ค.ศ 1984 ก็ทำให้การใช้มัลติโพรเซสซิงเพื่อเพิ่มความเร็วในการคำนวณแพร่หลายมากขึ้น เครื่องที่ขีดความสามารถสูงที่ถูกสร้างในปัจจุบันจะเป็นเครื่องประเภท shared - memory processor



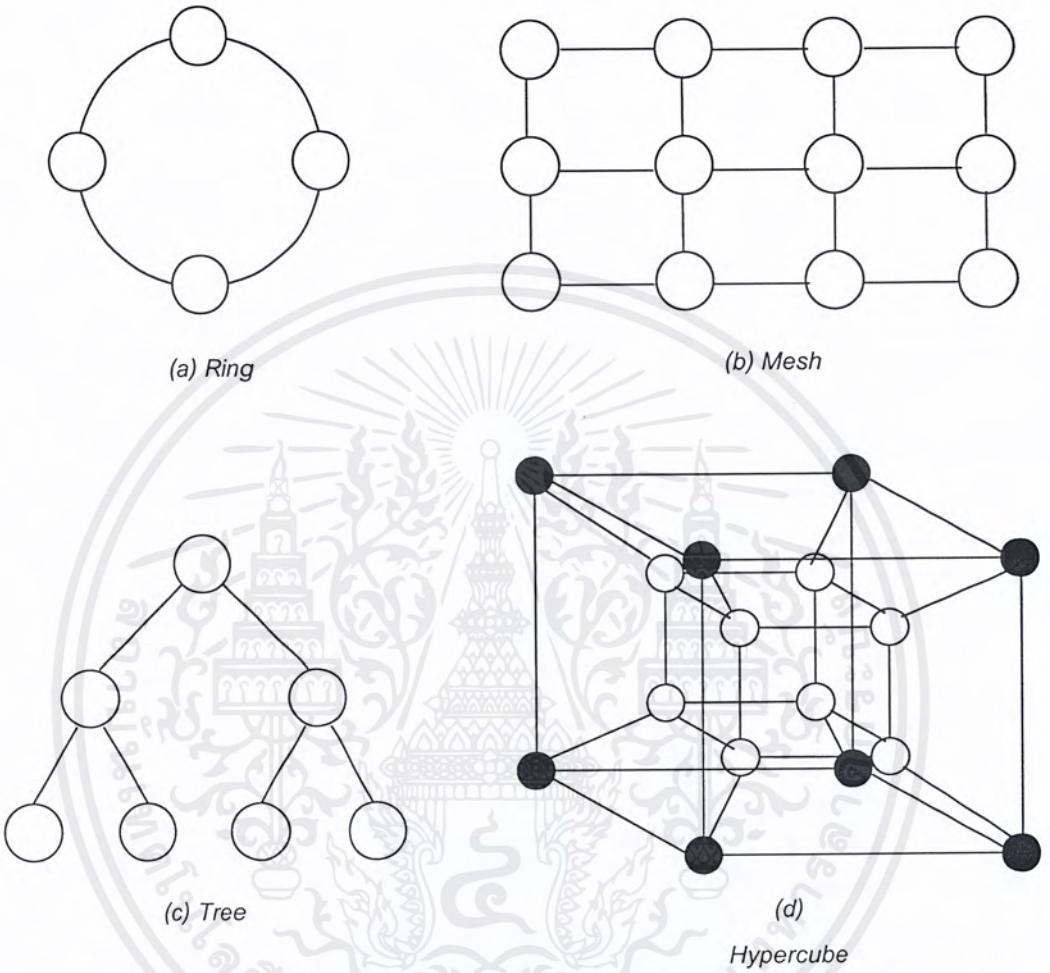
รูปที่ 2.1 แสดงซูเปอร์คอมพิวเตอร์ที่ใช้ Shared - memory หรือเรียกว่าเครื่องซูเปอร์คอมพิวเตอร์แบบ Multiprocessor.

- **Distributed – memory** แต่ละโพรเซสเซอร์จะใช้ส่วนของหน่วยความจำของตัวเอง การประมวลผลร่วมกันจะทำผ่านระบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน ดังรูปที่ 2.2 ในรูปแบบของเมสเซส (message passing)



รูปที่ 2. 2 แสดงซูเปอร์คอมพิวเตอร์ที่ใช้ Distributed - memory หรือเรียกว่าเครื่องซูเปอร์คอมพิวเตอร์แบบ Multicomputer.

เนื่องจากโพรเซสเซอร์ในระบบนี้ต้องติดต่อกันโดยใช้การแลกเปลี่ยนแมสเซส ดังนั้นกุญแจที่ช่วยในการออกแบบก็คือ รูปแบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน ที่จะต้องทำให้การปฏิบัติงานมีประสิทธิภาพตามที่ควรจะเป็น รูปแบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน เช่น 1.Ring 2.Mesh 3.Tree 4.Hypercube แสดงไว้ใน รูปที่ 2.3



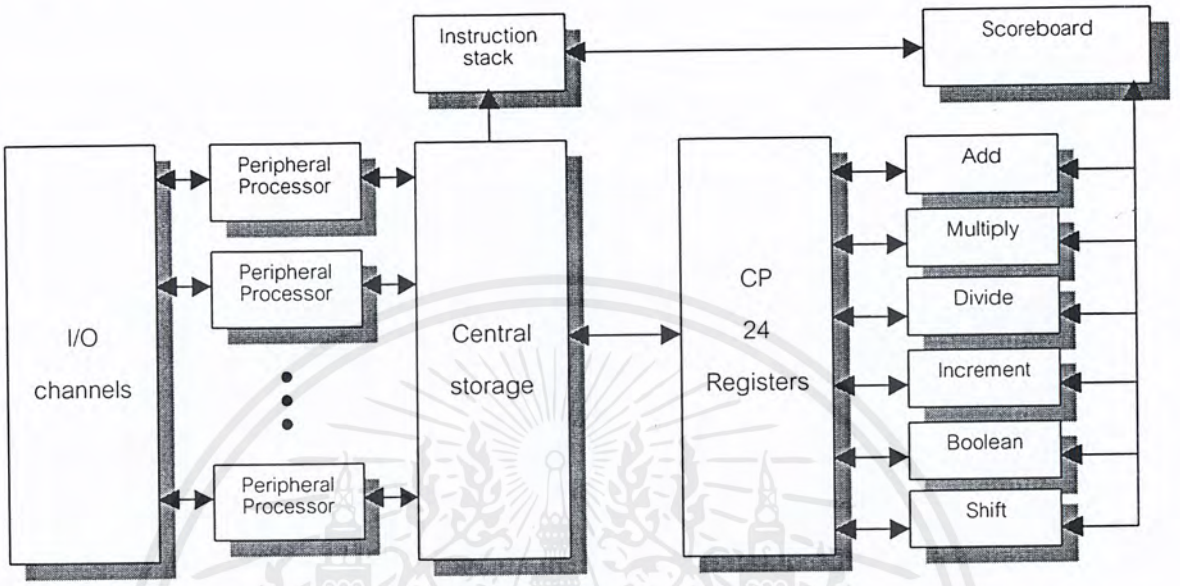
รูปที่ 2.3 แสดงรูปแบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน (Interconnection Network)

เครื่องคอมพิวเตอร์ประเภท Distributed – memory เขียนโปรแกรมได้ยากกว่าเครื่องคอมพิวเตอร์แบบ shared – memory อย่างไรก็ตามการเปิดตัวของ Cosmic cube ของ Caltech ในปี ค.ศ. 1980 ทำให้เกิดการใช้เครื่องแบบ distributed – memory กันมากขึ้น โดยเครื่องแบบ distributed – memory นี้เป็นเครื่องที่เหมาะกับปัญหาที่สามารถแยกออกเป็นส่วนย่อยที่ไม่ต้องใช้ร่วมกันบ่อย ๆ

โพรเซสเซอร์ที่มีระบบแตกต่างกันบางครั้งถูกนำมารวมในรูปแบบมัลติโพรเซสซิงเครื่องเดียวกัน เราเรียกเครื่องคอมพิวเตอร์ประเภทนี้ว่า " Heterogeneous multiprocessing " ใช้ประโยชน์ในการประมวลผลโปรแกรมที่ส่วนต่าง ๆ ของโปรแกรมต้องการการคำนวณที่แตกต่างกัน

2.2.2 มัลติฟังก์ชันโพรเซสเซอร์ (Multifunction processors) ในมัลติฟังก์ชันโพรเซสเซอร์กลุ่มของการปฏิบัติงานทางด้านการคำนวณและด้านตรรกะ จะแบ่งเป็นสับเซต โดยในแต่ละสับเซตจะมี

ตั้งแต่หนึ่งหน่วยฟังก์ชัน (Functional unit) ขึ้นไป ตัวอย่างเช่น หน่วยฟังก์ชันหนึ่งจะมีส่วนของ การคูณเลขทศนิยม (floating – point multiplication) , การหารเลขทศนิยม (floating – point division) และ การทำงานทางด้านตรรกะ โดยคุณสมบัติแบบขนานจะได้มาจากการทำงานของหน่วยฟังก์ชันที่เป็นแบบขนาน



รูปที่ 2.4 แสดงตัวอย่างของ functional units.

2.2.3 ไปป์ไลน์โพรเซสเซอร์ (Pipelined processors) การทำงานจะแบ่งเป็นหลาย ๆ สเตจ ตัวอย่างเช่น คำสั่งการคูณเลขทศนิยม แต่ละสเตจจะแบ่งการทำงานออกเป็นส่วนย่อย ๆ เช่น adding the exponents, multiplying the mantissas, rounding และ normalizing the result แต่ละสเตจจะต่อเนื่องกันเป็นเส้นตรง โดยการทำงานแบบไปป์ไลน์จะผ่านเข้าสู่จุดแรกและกระทำต่อไปจากสเตจหนึ่งไปยังสเตจต่อไปจนกระทั่งจบการทำงาน ผลลัพธ์จะออกมาทางอีกด้านหนึ่งของไปป์ไลน์ และจะสามารถเข้าถึงคุณสมบัติแบบขนานได้โดยการประมวลผลหลาย ๆ ไปป์ไลน์พร้อมกัน โดยทั่วไปไปป์ไลน์จะทำงานแบบซิงโครไนซ์ ในช่วงเวลาหนึ่งการทำงานในสเตจต่าง ๆ จะเคลื่อนย้ายไปพร้อม ๆ กันยังสเตจต่อไป และสเตจแรกจะรับการทำงานใหม่เข้ามา ในช่วงเวลาระหว่างการเคลื่อนย้ายการทำงานแต่ละครั้งเรียกว่า " cycle time " ของโพรเซสเซอร์

พิจารณาไปป์ไลน์ที่มี S สเตจ และต้องทำ M โอเปอเรชัน ไปป์ไลน์ต้องการ S ไซเคิลเพื่อทำงาน หลังจากนั้นผลลัพธ์ 1 ตัวจะออกมาทุก ๆ ไซเคิล ถ้าเวลาในแต่ละไซเคิล(cycle time) เท่ากับ T เวลาทั้งหมดที่ต้องใช้ทำ M โอเปอเรชัน จะเท่ากับ (S + M) * T ถ้าไม่ใช้ไปป์ไลน์ต้องใช้เวลาทั้งหมดเท่ากับ S * M * T ประสิทธิภาพสูงสุดของไปป์ไลน์ยูนิทเท่ากับ 1 / T operation per second (ถ้าใช้เพียง 1 ไปป์ไลน์) ในคอมพิวเตอร์บางเครื่องมีการใช้หลายไปป์ไลน์ทำให้สามารถประมวลผลโอเปอชั่นได้พร้อม ๆ กันในแต่ละสเตจ หน่วยฟังก์ชันที่มี N ไปป์ไลน์ จะมีประสิทธิภาพสูงสุดเท่ากับ N / T operation per second

ไปป์ไลน์มักใช้ในหน่วยควบคุม (control unit) ของโพรเซสเซอร์ ตัวอย่างเช่น instruction fetch, instruction decode, fetching operands, execution operation โดยในปัจจุบันหน่วยควบคุมของโพรเซสเซอร์ส่วนมากจะใช้ไปป์ไลน์

ปัญหาหลักที่มีผลต่อการทำงานในระบบไปป์ไลน์โพรเซสซิง คือ

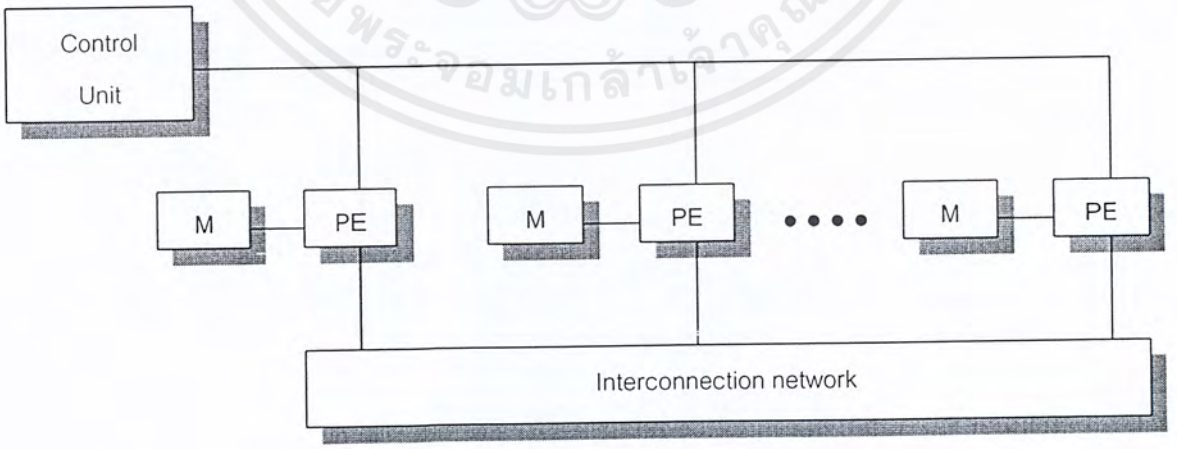
- *ซิงโครไนซ์เซชัน (Synchronization)* แต่ละสเตจในไปป์ไลน์ต้องใช้เวลาในการทำงานในสำเร็จเท่ากัน เพื่อทำให้งานสามารถไหลไปได้ระหว่างสเตจโดยไม่หยุดชะงักที่สเตจใดสเตจหนึ่ง ถ้าเวลาที่ใช้ในสเตจหนึ่งเท่ากับ t_1 และเวลาที่สเตจสองใช้เท่ากับ t_2 ซึ่ง t_1 น้อยกว่า t_2 ดังนั้นงานจะต้องรอเป็นเวลา (t_2 - t_1) ก่อนที่จะเข้าสเตจสองได้ โดยงานแบบนี้ต้องการที่เก็บข้อมูลชั่วคราวระหว่างสเตจ (เราเรียกว่า บัฟเฟอร์)

- *ฟองอากาศในไปป์ไลน์ (Bubbles in Pipeline)* ถ้าทาส์กบางทาส์กไม่มีในงาน ก็จะทำให้มีโพรเซสเซอร์ที่ว่างงานเกิดขึ้น

- *การทนทานต่อความผิดพลาด (Fault Tolerance)* ระบบไม่มีความทนทานต่อความผิดพลาด ถ้ามีสเตจหนึ่งในไปป์ไลน์ผิดพลาดขึ้นมา จะทำให้ระบบไปป์ไลน์หยุดชะงักได้

- *การสื่อสารระหว่างทาสก์ (Intertask Communication)* เวลาที่ใช้ในการส่งงานระหว่างสเตจควรจะน้อยกว่าเวลาที่ใช้ในการทำงาน

2.2.4 อาร์เรย์โพรเซสเซอร์ (Array Processors) ประกอบด้วยหลาย ๆ หน่วยของการคำนวณและตรรกะ ซึ่งเรียกว่า Processing Elements (PEs) หน่วยควบคุมของโพรเซสเซอร์จะส่งคอมมาน ซึ่งสอดคล้องกับแต่ละคำสั่งไปที่ทุก ๆ PEs ในกรณีนี้คำสั่งแบบเวกเตอร์ (Vector Instruction) สามารถเอ็กซคิวโดยการทำโอเปอเรชันเดียวกันพร้อม ๆ กันที่ array elements ต่าง ๆ หน่วยความจำจะแบ่งเป็นของแต่ละ PEs ซึ่งสามารถเชื่อมต่อกันได้ผ่านระบบการเชื่อมต่อเน็ตเวิร์กระหว่างกัน เพื่อจุดประสงค์ในการจัดการกับข้อมูลก่อนการเอ็กซคิวของ array operation เราสามารถเพิ่มคุณสมบัติแบบขนานได้โดยเพิ่มจำนวนของ PEs



รูปที่ 2.5 แสดง Array processor.

ชนิดของคอมพิวเตอร์	รูปแบบของการทำงานแบบขนาน	ขนาดของทาส์ก	ความง่ายในการเขียนโปรแกรม
ไปป์ไลน์	Temporal Parallelism	เล็ก	ค่อนข้างง่าย
อาร์เรย์โพรเซสเซอร์	Data Parallelism	เล็ก	ค่อนข้างง่าย
Shared – memory (ประมาณ 12 –16 โพรเซสเซอร์)	Multiple tasks and multiple data	กลาง	ขึ้นอยู่พื้นที่ที่สามารถใช้งานได้ และ ค่อนข้างง่าย
Distributed – memory	Multiple tasks and multiple data	ใหญ่	ยาก แต่มีการจัดเนื้อที่ให้ทาส์กดี และ ต้องการการซิงโครไนซ์เซชันด้วย

ตารางที่ 2.1 แสดงการเปรียบเทียบรูปแบบการทำงานในระบบซูเปอร์คอมพิวเตอร์แบบต่าง ๆ

2.3 การประมวลผลแบบขนาน (Parallel Processing)

คอมพิวเตอร์โดยทั่วไปนั้นเป็นเครื่องจักรที่ทำงานเป็นลำดับขั้นตอนก่อนหลัง (Sequential) การโปรแกรมภาษาส่วนมากจึงต้องการให้ผู้เขียนโปรแกรมสร้างโปรแกรมที่เป็นลำดับขั้นตอน เพื่อแก้ปัญหาการทำงานที่ต้องเป็นลำดับขั้นตอนจึงให้กำเนิดสัญญาณควบคุมพร้อมกันหลายสัญญาณ เพื่อให้เอ็กซคิวหลาย คำสั่งพร้อมกันได้เรียกว่า Parallel เมื่อเทคโนโลยีคอมพิวเตอร์ได้พัฒนาขึ้น และราคาของฮาร์ดแวร์ลดลง ผู้ออกแบบจึงมีโอกาสในด้านการทำงานแบบขนานมากขึ้น

การจัดการแบบขนานนั้นมีวิธีการที่เด่น ๆ 3 แบบ คือ

- Multiprocessing
- Parallel processor
- Vector computation

2.3.1 มัลติโพรเซสซิง วิธีการแบบนี้โปรแกรมจะแบ่งออกเป็นส่วนย่อย ๆ และโพรเซสเซอร์แต่ละตัวก็จะสามารถเอ็กซคิวโปรแกรมได้พร้อมกัน ระบบมัลติโพรเซสซิงสามารถแบ่งได้เป็น 2 แบบ คือ

- Shared – memory โพรเซสเซอร์แต่ละตัวจะใช้หน่วยความจำร่วมกัน โดยผ่านทางการเชื่อมต่อเน็ตเวิร์กระหว่างกัน
- Distributed – memory โพรเซสเซอร์แต่ละตัวจะมีหน่วยความจำเป็นของตัวเอง แล้วประมวลผลร่วมกันผ่านทางเชื่อมต่อเน็ตเวิร์กระหว่างกัน

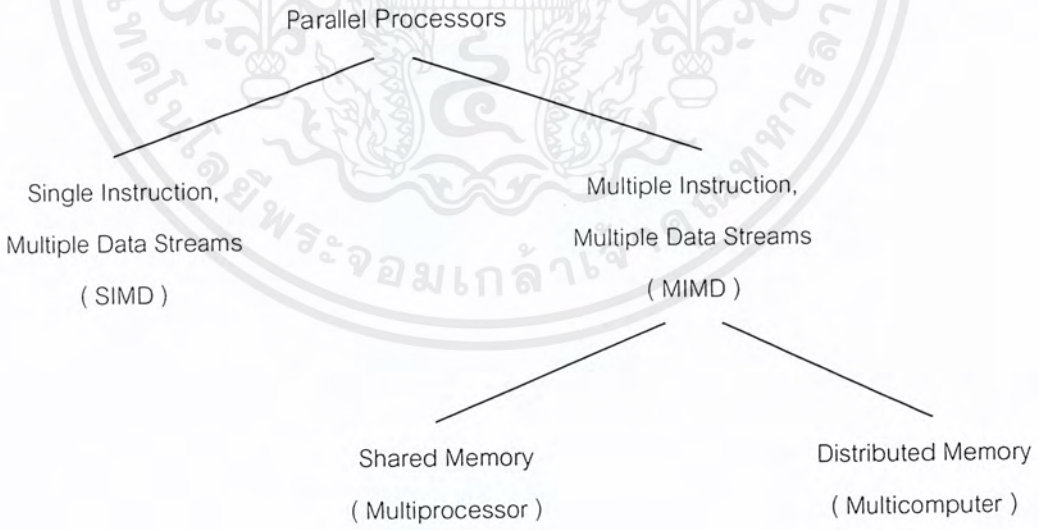
2.3.2 โพรเซสเซอร์แบบขนาน สามารถจัดแบ่งโดยยึด instruction stream และ data stream เป็นหลักได้เป็น 4 ประเภท ดังนี้

- **Single Instruction Single Data (SISD) stream** แบบนี้โพรเซสเซอร์หนึ่งตัวจะแปลคำสั่งหนึ่ง instrution stream เพื่อทำงานกับข้อมูลที่เก็บไว้ในหน่วยความจำหน่วยหนึ่ง

- **Single Instruction Multiple Data (SIMD) stream** แบบนี้โพรเซสเซอร์ทั้งหมดจะประมวลผลคำสั่งที่เหมือนกันบนข้อมูลที่ต่างกัน ซึ่งเป็นการเพิ่มประสิทธิภาพและความเร็วขึ้น โดยใช้ประโยชน์จากการทำคาต้าพาราลเลลิซึม แต่ยังมีข้อจำกัดอยู่ในการใช้งานบางลักษณะเท่านั้น แบ่งเป็น 1.เวกเตอร์โพรเซสเซอร์ และ 2.อาร์เรย์โพรเซสเซอร์

- **Multiple Instruction Single Data (MISD) stream** แบบนี้ข้อมูลที่เป็นแบบเรียงลำดับจะถูกส่งไปให้ชุดของโพรเซสเซอร์ โดยที่โพรเซสเซอร์แต่ละตัวจะเอ็กซคิวคำสั่งที่แตกต่างกันไปตามลำดับ รูปแบบนี้ยังไม่เคยได้รับการนำไปใช้งานจริงเลย

- **Multiple Instruction Multiple Data (MIMD) stream** แบบนี้โพรเซสเซอร์แต่ละตัวอาจประมวลผลคำสั่งที่แตกต่างกันบนข้อมูลที่ต่างกันในเวลาเดียวกันได้ ทำให้มีความสามารถที่สูง เช่น ความน่าเชื่อถือ คือ ถ้าโพรเซสเซอร์ตัวใดทำงานผิดพลาด งานที่ทำอยู่ก็สามารถทำต่อไปได้โดยโพรเซสเซอร์ตัวอื่น ๆ แต่ก็ยังมีปัญหาบางอย่าง เช่น การจัดแบ่งงานของแต่ละโพรเซสเซอร์ , การซิงโครไนซ์เซชันระหว่างโพรเซสเซอร์แต่ละตัว แต่ก็ยังมีปัญหาเกิดขึ้น เช่น การจัดแบ่งงานของโพรเซสเซอร์ , การซิงโครไนซ์เซชันระหว่างโพรเซสเซอร์แต่ละตัว



รูปที่ 2.6 แสดงการแบ่งชนิดของ Parallel Processors

2.3.3 การคำนวณแบบเวคเตอร์ (Vector Computation)

โดยทั่วไปปัญหาที่มีลักษณะเฉพาะ ที่ต้องการความเร็วสูง ความแม่นยำ และโปรแกรมที่การประมวลผลทางตัวเลขเป็นจำนวนทศนิยมไม่รู้จบบนอาร์เรย์ ปัญหาส่วนมากจะถูกเรียกว่า Continuous - field simulation ตัวอย่างเช่น ปรากฏการณ์ทางฟิสิกส์ ที่แสดงพื้นผิว หรือ พื้นที่สามมิติ (การไหลของอากาศที่ใกล้กับพื้นผิวของเครื่องบิน) พื้นผิวเหล่านี้จะถูกประมาณเป็นส่วนย่อยๆ ของพื้นผิว เซตของ สมการ Differential จะกำหนดพฤติกรรมทางฟิสิกส์ของแต่ละส่วนบนพื้นผิว โดยสมการจะถูกแทนที่เป็นค่าของอาร์เรย์, ค่าสัมประสิทธิ์ และการแก้ปัญหาที่เกี่ยวข้องกับโอเปอเรชันทางคณิตศาสตร์ที่ซ้ำไปซ้ำมาบนข้อมูลของอาร์เรย์ เพื่อแก้ปัญหาดังที่กล่าวมาจึงมีการพัฒนาซูเปอร์คอมพิวเตอร์ขึ้น

อาร์เรย์โพรเซสเซอร์ คือ การออกแบบที่ใช้สำหรับการคำนวณแบบเวคเตอร์ แม้ว่าซูเปอร์คอมพิวเตอร์ จะเหมาะสมสำหรับการคำนวณแบบเวคเตอร์ มันก็ยังสามารถทำงานแบบคอมพิวเตอร์ทั่วไปได้นั่นก็คือ การประมวลผลแบบสเกลาร์ และการประมวลผลข้อมูลทั่วๆไป โดยอาร์เรย์โพรเซสเซอร์จะไม่รวมถึงการประมวลผลแบบสเกลาร์

2.3.4 การคำนวณทางเวคเตอร์ (Approaches to Vector Computation)

หลักการออกแบบซูเปอร์คอมพิวเตอร์ หรือ อาร์เรย์โพรเซสเซอร์ จะต้องตระหนักถึงงานหลักที่จะแสดงการทำงานทางคณิตศาสตร์บนอาร์เรย์ หรือ อาร์เรย์ของจำนวนทศนิยมไม่รู้จบ ในคอมพิวเตอร์โดยทั่วไปนั้นจะต้องการการทำซ้ำในแต่ละสมาชิกของอาร์เรย์ ตัวอย่างเช่น พิจารณาเวคเตอร์สองเวคเตอร์ (อาร์เรย์ 1 มิติ) A และ B เราจะบวกแต่ละตัวแล้วได้ผลลัพธ์เป็น C ดังรูปที่ 7 การกระทำนั้นจะต้องการการบวกแยกจากกัน 6 ครั้ง เราจะเพิ่มความเร็วในการคำนวณได้อย่างไร คำตอบก็คือ การใช้เทคนิคการทำแบบขนาน (Parallel)

$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 59.7 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.8 \\ 1006.903 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

รูปที่ 2. 7 แสดงตัวอย่างของการบวกเวคเตอร์

การเข้าถึงแบบต่างๆจะทำได้ผลเมื่อคำนวณเวคเตอร์ในแบบขนาน เราจะสาธิตด้วยตัวอย่างสักหนึ่งตัวอย่าง พิจารณาการคูณเวคเตอร์ C = A×B เมื่อ A, B, C เป็นเมทตริกซ์ N×N สูตรของ C ก็คือ

$$C_{i,j} = \sum_{K=1}^{N} a_{i,k}\, b_{k,j}$$

จากรูป 2.8 แสดงโปรแกรม FORTRAN สำหรับการคำนวณที่รันบนโพรเซสเซอร์แบบสเกลาร์ธรรมดา อีกวิธีที่พัฒนาขึ้นเรียกว่า เวคเตอร์โพรเซสเซอร์ ซึ่งวิธีการนี้เป็นไปได้ที่จะจัดการข้อมูลแบบเวคเตอร์ 1 มิติได้ จากรูปที่ 2.9 คือโปรแกรมภาษา FORTRAN ซึ่งมีรูปแบบคำสั่งใหม่ที่อนุญาตให้คำนวณเวคเตอร์ที่ถูกเจาะจงโดยใช้สัญลักษณ์ J = 1, N

```
      DO 100 I = 1,N
      DO 100 J =1,N
      C(I,J) = 0,0
      DO 100 K = 1,N
      C(I,J) = C(I,J)+ A(I,K)*B(K,J)
100   CONTINUE
```

รูปที่ 2.8 แสดง Scalar Processing.

```
      DO 100 I = 1,N
      DO 100 J =1,N
      C(I,J) = 0,0
      DO 100 K = 1,N
      C(I,J) = C(I,J)+ A(I,K)*B(K,J)
100   CONTINUE
```

รูปที่ 2.9 แสดง Scalar Processing.

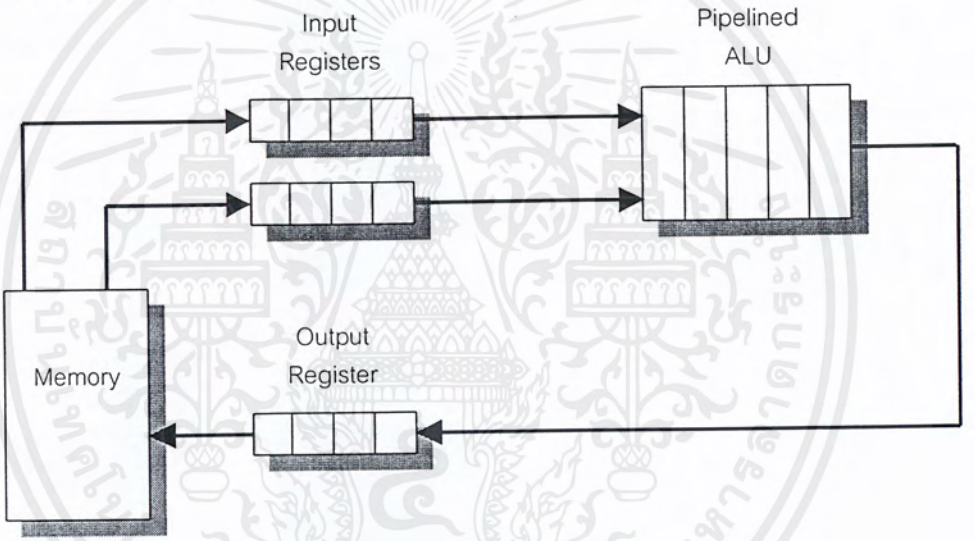
```
DO 50 J = 1,N
      FORK 100
50    CONTINUE
      J = N
100   DO I = 1,N
      C(I,J) = 0,0
      DO 200 K = 1,N
      C(I,J) = C(I,J)+ A(I.K)*B(K,J)
200   CONTINUE
      JOIN N
```

รูปที่ 2.10 แสดง Parallel Processing.

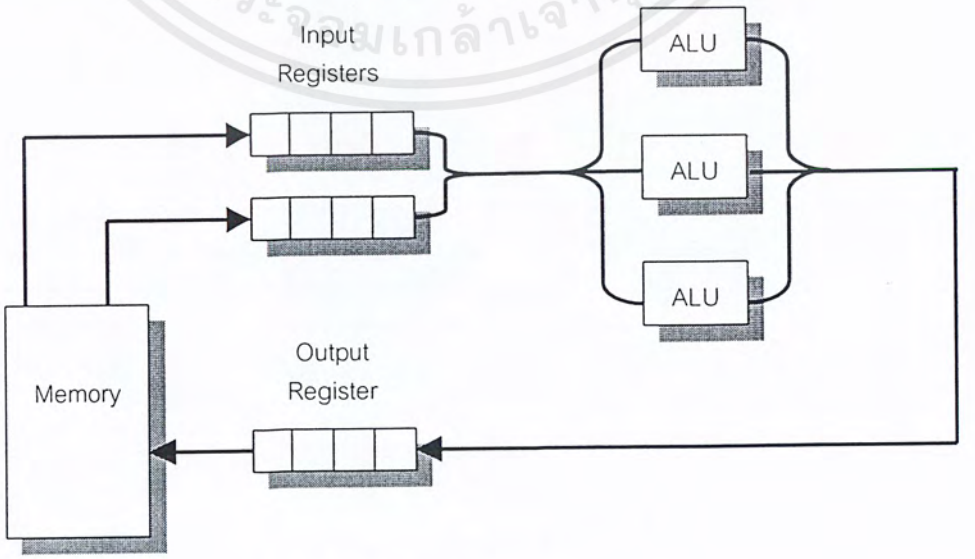
จากตัวอย่างที่ผ่านมาเป็นการเข้าถึงการคำนวณแบบเวคเตอร์ในการเขียนโปรแกรม ลองพิจารณาถึงการจัดการของโพรเซสเซอร์ซึ่งการจัดการภายในนั้นมีวิธีเด่นๆ 3 วิธี คือ

- Pipelined ALU
- Parallel ALU
- Parallel processor

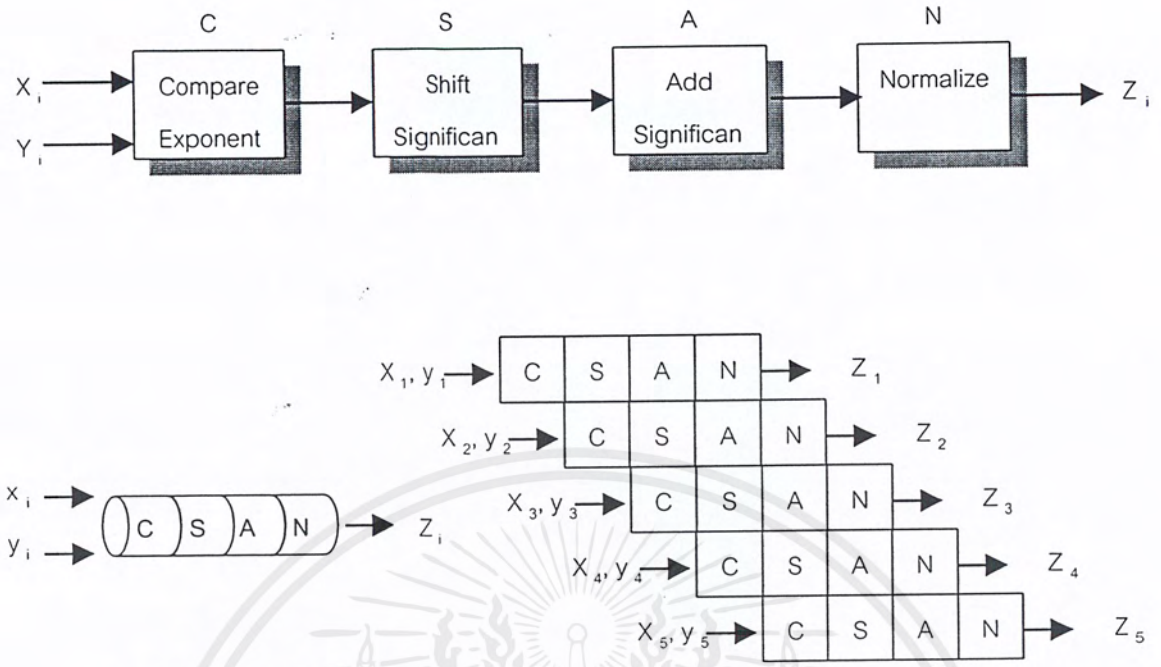
รูปที่ 2.12 และ รูปที่ 2.13 แสดงถึง Pipelined ALU และ Parallel ALU ตามลำดับ รูปแบบก็คือการยืดการทำงานของ ALU ออกไป เนื่องจากการคำนวณจำนวนทศนิยมซ้ำไม่รู้จบค่อนข้างสลับซับซ้อน จึงมีโอกาสที่จะแยกการคำนวณจำนวนทศนิยมไม่รู้จบนั้นได้เป็นหลายๆขั้นตอน (stages) เพื่อเพิ่มประสิทธิภาพการทำงานของไปป์ไลน์ การคำนวณแบบเวคเตอร์นั้นไม่ควรที่จะต้องเข้าถึงหน่วยความจำหลักในการคำนวณ ไปป์ไลน์ ควรจะมีรีจิสเตอร์สำหรับเก็บข้อมูลทางเวคเตอร์โดยตรงซึ่งจะเรียกว่า *"เวคเตอร์รีจิสเตอร์ (Vector Register)"* ในรูปที่ 2.12 และ 2.13 Vector operand ในแต่ละส่วนจะถูกโหลดเป็นบล็อคเข้าสู่เวคเตอร์รีจิสเตอร์ โดยผลลัพธ์จะถูกเก็บไว้ในเวคเตอร์รีจิสเตอร์ ด้วย



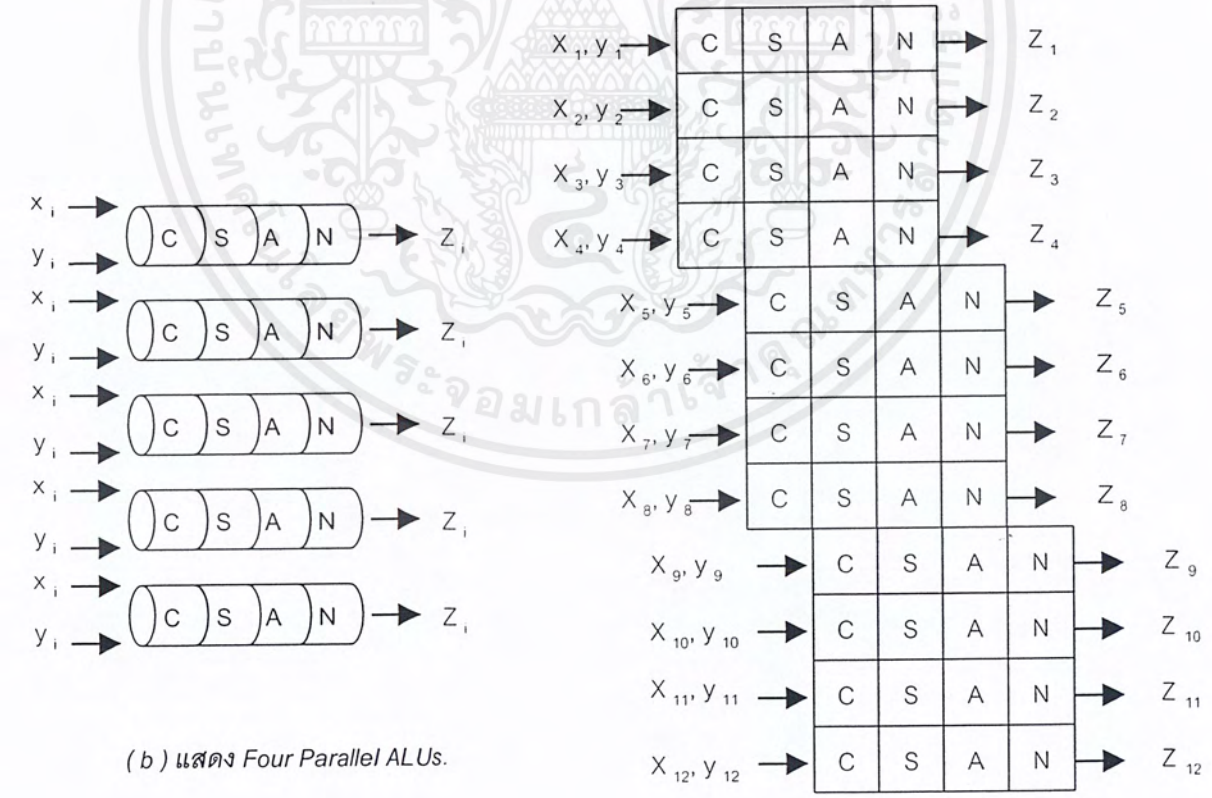
รูปที่ 2.11 แสดง Pipelined ALU.



รูปที่ 2.12 แสดง Parallel ALUs.



(a) แสดง Pipelined ALU.



(b) แสดง Four Parallel ALUs.

รูปที่ 2.13 แสดงการประมวลผลแบบไปป์ไลน์

2.3.5 เชนนิ่ง (Chainning)

เป็นวิธีการหนึ่งที่ใช้ในการเพิ่มความเร็วในการประมวลผลซึ่งพบได้ในซูเปอร์คอมพิวเตอร์ Cray กฎพื้นฐานของการเชนนิ่ง คือ การประมวลผลทางเวคเตอร์จะเริ่มที่ส่วนแรกของข้อมูลทางเวคเตอร์ที่ว่างอยู่ และหน่วยของฟังก์ชันการทำงาน (เช่น บวก ลบ คูณ หาร) ของมันนั้นต้องเป็นอิสระ จุดสำคัญก็คือ เชนนิ่งจะทำให้ผลลัพธ์ที่ออกมาจากหน่วยของฟังก์ชันการทำงานหนึ่ง และจะถูกดึงเข้าไปในหน่วยของฟังก์ชันการทำงานอื่นทันที ดังนั้นเมื่อเวคเตอร์รีจิสเตอร์ถูกใช้ ผลลัพธ์ที่ได้ไม่ต้องนำไปเก็บในหน่วยความจำทันที สามารถนำไปใช้ได้ก่อนที่การทำงานทางเวคเตอร์ จะถูกสร้างให้รันได้สมบูรณ์

ตัวอย่างเช่น เมื่อคำนวณ C = (s × A) + B เมื่อ A, B, C เป็นเวคเตอร์ และ s เป็นสเกลาร์ ถ้าเป็นซูเปอร์คอมพิวเตอร์ Cray จะเอ็กซคิว 3 คำสั่งทันที ส่วนต่างๆจะเข้าสู่การทำงานไปป์ไลน์หลายๆ ตัวทันที โดยผลลัพธ์จะถูกส่งไปที่ไปป์ไลน์แอดเดอร์ (Pinelined Adder) และผลบวกจะถูกแทนที่ในเวคเตอร์รีจิสเตอร์ ทันทีที่ทำการบวกเสร็จ

1. โหลดเวคเตอร์ A → Vector register(VR1)
2. โหลดเวคเตอร์ B → VR2
3. คูณเวคเตอร์ s×VR1 → VR3
4. บวกเวคเตอร์ VR3+VR2 → VR4
5. เก็บค่าเวคเตอร์ VR4 → C

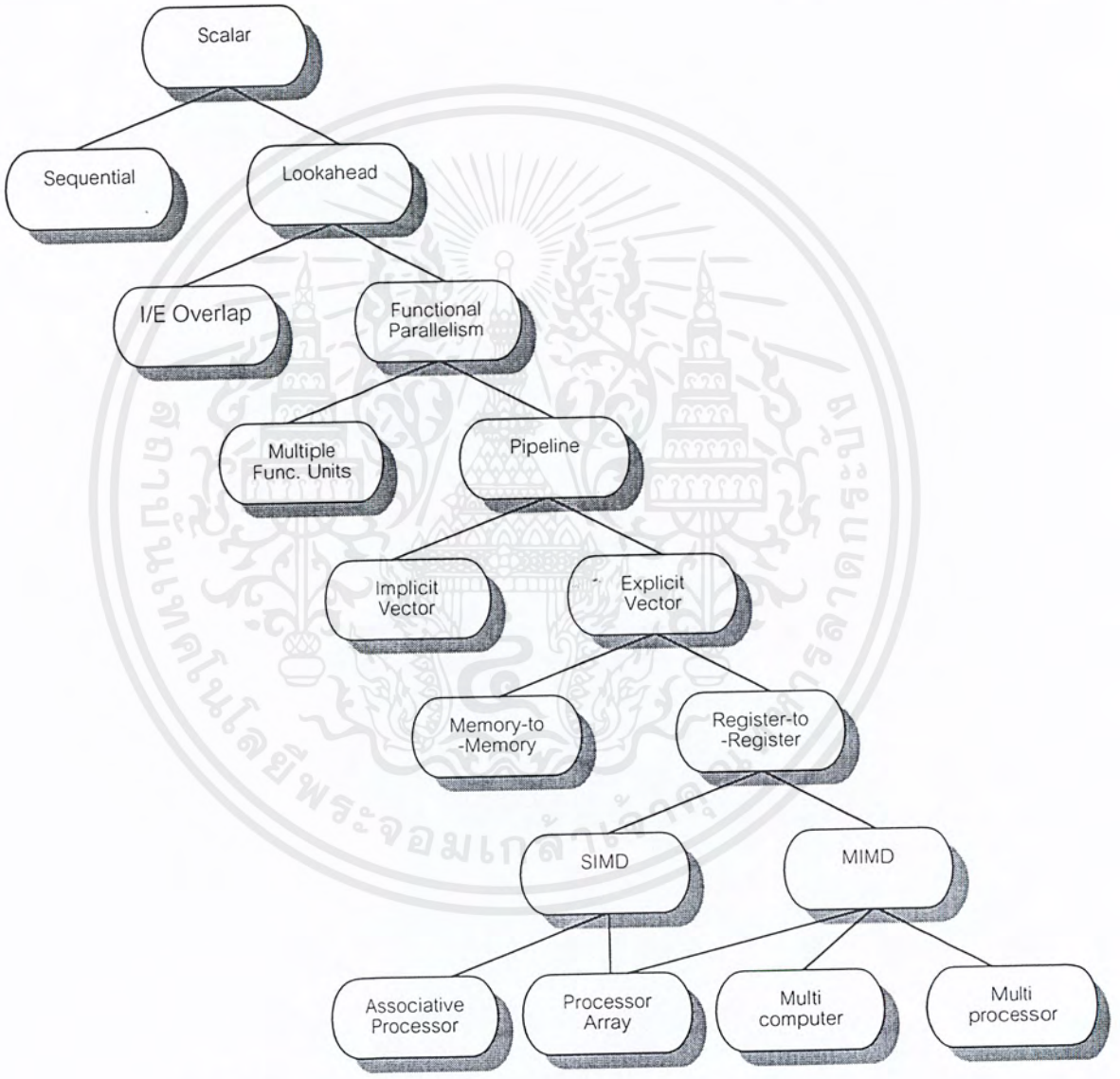
จากการคำนวณทางเวคเตอร์ด้านบน คำสั่งที่ 2 และ 3 สามารถ chained(pipelined) ได้เนื่องจากตำแหน่งในหน่วยความจำนั้นแตกต่างกัน และใช้รีจิสเตอร์คนละตัวกัน ส่วนคำสั่งที่ 4 ต้องการผลลัพธ์ของคำสั่งที่ 2 และ 3 แต่มันสามารถ chain ได้ด้วย ถ้าเมื่อใดที่ส่วนแรกของเวคเตอร์รีจิสเตอร์ว่าง การทำงานในคำสั่งที่ 4 ก็จะเริ่มได้

วิธีอื่นอีกวิธีที่ใช้สร้างการประมวลผลแบบเวกเตอร์ ก็มีโดยการใช้ ALU หลายๆตัวในโพรเซสเซอร์ตัวเดียวภายใต้การควบคุมของหน่วยควบคุมเดียว ในกรณีนี้หน่วยควบคุมจะวางเส้นทางของข้อมูลไปสู่ ALU ดังนั้นจึงสามารถทำงานเป็นแบบขนานได้ เราสามารถที่จะใช้ไปป์ไลน์ในแต่ละ ALU ที่ต่อแบบขนานกัน ดังแสดงในรูปที่ 2.14(b) เป็นกรณีที่ใช้ ALU 4 ตัว ต่อกันแบบขนาน

2.4 สถาปัตยกรรมของระบบคอมพิวเตอร์

การศึกษาเกี่ยวกับสถาปัตยกรรมคอมพิวเตอร์เกี่ยวข้องทั้งระบบของฮาร์ดแวร์ และ ความต้องการของโปรแกรมและซอฟท์แวร์ ดังจะเห็นได้จากการเขียนโปรแกรมด้วยภาษาแอสแซมบลี้ สถาปัตยกรรมคอมพิวเตอร์ถูกแสดงได้จากชุดคำสั่งของมันซึ่งรวมถึง ออปโค้ด (opcode), วิธีการอ้างถึงแอดเดรส (addressing mode), รีจิสเตอร์, หน่วยความจำเสมือน (virtual memory) และอื่นๆอีก สรุปคือการศึกษาสถาปัตยกรรมคอมพิวเตอร์นั้นจะรวมถึงการศึกษาทั้ง สถาปัตยกรรมของชุดคำสั่ง และ ระบบการทำงานของเครื่อง

ในรอบ 40 ปีที่ผ่านมาสถาปัตยกรรมคอมพิวเตอร์มีการเจริญเติบโตมากกว่าที่จะเกิดการเปลี่ยนแปลง (รูปที่ 11) เราเริ่มจากสถาปัตยกรรมคอมพิวเตอร์ของ Von Neuman เป็นเครื่องคอมพิวเตอร์แบบที่ทำงานตามลำดับที่เอ็กซคิวข้อมูลแบบสเกลาร์ คอมพิวเตอร์แบบทำงานตามลำดับถูกพัฒนาขึ้นจากการทำงานแบบ bit – serial เป็น word – parallel และจากการทำงานกับเลขจำนวนเต็ม (fix point) เป็นการทำงานกับเลข ทศนิยม (floating point) สถาปัตยกรรมคอมพิวเตอร์ของ Von Neuman ทำงานช้าเนื่องจากข้อกำหนดของการทำงานแบบตามลำดับ



คำอธิบาย:

I/E: Instruction Fetch and Execute.

SIMD: Single Instruction stream and Multiple Data streams.

MIMD: Multiple Instruction streams and Multiple Data streams.

รูปที่ 2.14 แสดงวิวัฒนาการของสถาปัตยกรรมคอมพิวเตอร์

2.4.1 Lookahead, การทำงานแบบขนาน และไปป์ไลน์

เทคนิค Lookahead จะทำการดึงคำสั่งมาก่อนล่วงหน้า ในกรณีการทำงานแบบ overlap I/E (instruction fetch/ decode and execution) และใช้ฟังก์ชันการทำงานแบบขนาน ซึ่งคุณสมบัติแบบขนานของฟังก์ชันการทำงานจะถูกสนับสนุนโดย

1. ใช้หน่วยของฟังก์ชันการทำงานหลาย ๆ หน่วยที่ทำงานพร้อมกัน
2. การใช้ไปป์ไลน์ในระดับการประมวลผลต่าง ๆ กัน

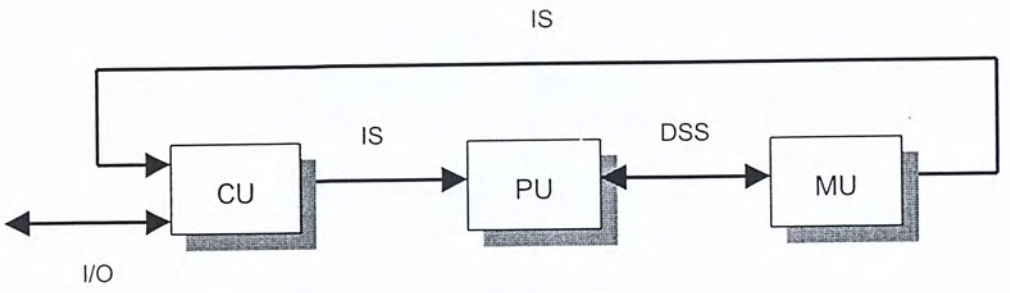
ต่อมาได้มีการเอ็กซคิวโดยใช้ไปป์ไลน์, การคำนวณทางคณิตศาสตร์โดยใช้ไปป์ไลน์ และการทำงานที่เกี่ยวกับการเข้าถึงหน่วยความจำ

2.4.2 Explicit vector instructions ถูกนำเสนอพร้อมกับการปรากฏของเวคเตอร์โพรเซสเซอร์ โดยเวคเตอร์โพรเซสเซอร์ถูกสร้างขึ้นจากการใช้เวคเตอร์ไปป์ไลน์หลายๆอันที่สามารถทำงานได้พร้อมกัน โพรเซสเซอร์แบบเวคเตอร์ไปป์ไลน์สามารถแบ่งได้เป็น 2 ประเภท คือ

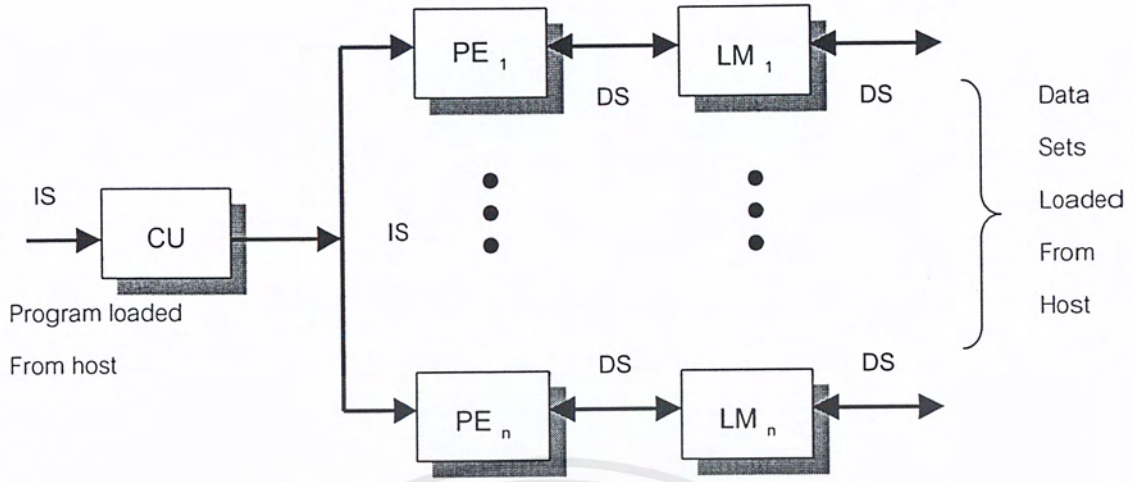
1. *Memory – to – memory architecture* แบบนี้ vector operand จะถูกส่งจากหน่วยความจำผ่านไปยังไปป์ไลน์ และกลับมาสู่หน่วยความจำเหมือนเดิม

2. *Register – to – register architecture* แบบนี้จะมีเวคเตอร์รีจิสเตอร์เป็นตัวกลางอยู่ระหว่างหน่วยความจำ และ ไปป์ไลน์ของฟังก์ชันการทำงาน

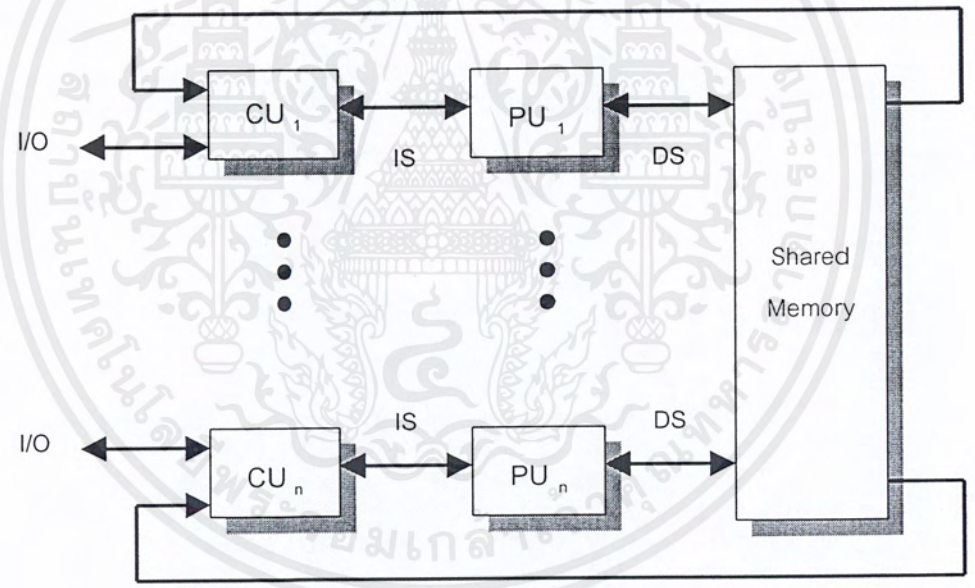
2.4.3 Flynn classification ปี 1972 Michael Flynn เสนอการจัดแบ่งประเภทที่หลากหลายของสถาปัตยกรรมคอมพิวเตอร์โดยยึด instruction stream และ data stream เป็นหลัก (รูปที่ 2.15) แสดงเครื่องที่ทำงานแบบตามลำดับที่ถูกเรียกว่า **SISD (single instruction stream over a single data stream)** computers เวคเตอร์คอมพิวเตอร์ถูกสร้างขึ้นด้วยส่วนประกอบที่ทำงานแบบสเกลาร์ และเวคเตอร์จะอยู่ในพวก **SIMD (single instruction stream over multiple data streams)** (รูปที่ 2.16) คอมพิวเตอร์แบบขนานจะอยู่ในรูปแบบของ **MIMD (multiple instruction streams over multiple data streams)** (รูปที่ 2.17 และ รูปที่ 2.18)



รูปที่ 2.15 แสดง SISD uniprocessor architecture



รูปที่ 2.16 แสดง SIMD architecture (with distributed memory)



รูปที่ 2.17 แสดง MIMD architecture (with shared memory)

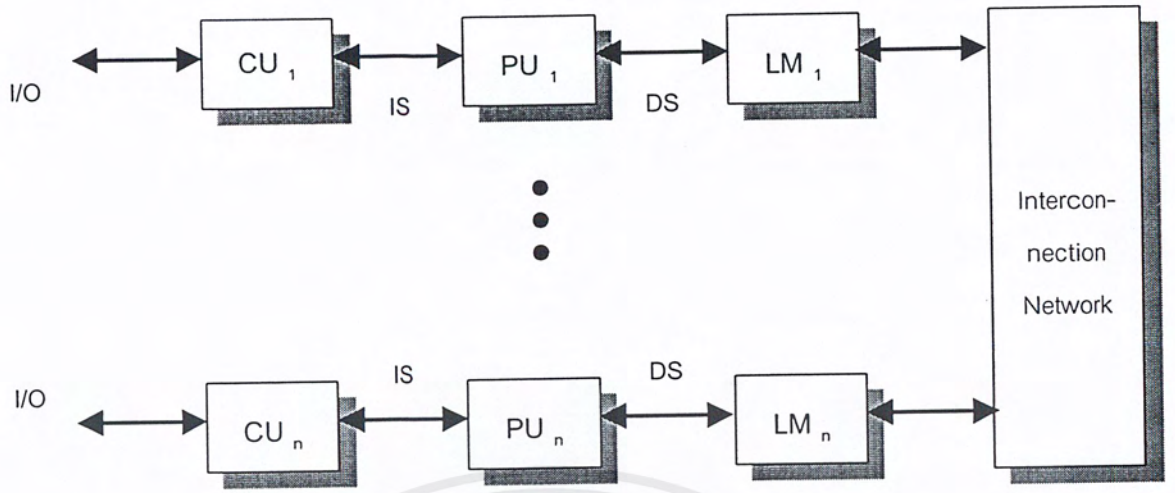
คำอธิบาย:

CU = Control Unit PU = Processing Unit

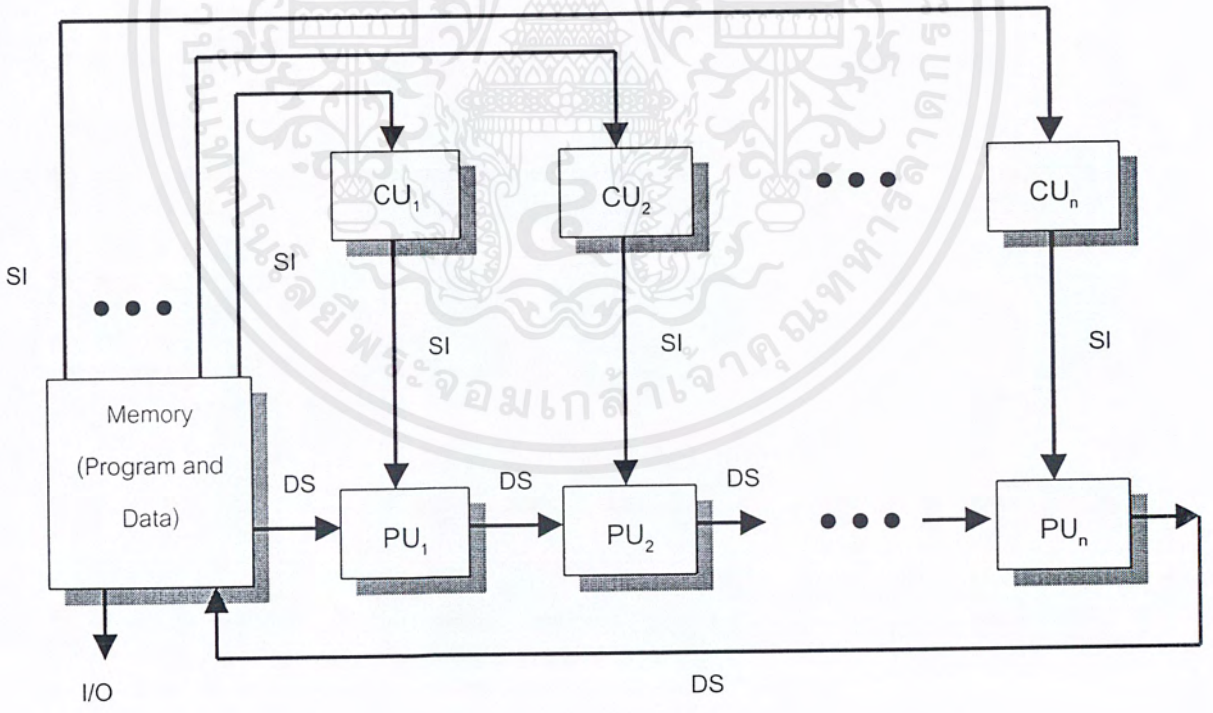
MU = Memory Unit *IS = Instruction Stream*

DS = Data Stream *PE = Processing Element*

LM = Local Memory



รูปที่ 2.18 แสดง MIMD architecture (with distributed memory)



รูปที่ 2.19 แสดง MISD architecture (the systolic array)

ส่วน **MISD (multiple instruction streams and a single data stream)** (รูปที่ 2.20) สายของข้อมูลเดินทางผ่านโพรเซสเซอร์ที่เรียงต่อกันเป็นเส้นตรง และทำการเอ็กซคิวคำสั่งที่แตกต่างกัน สถาปัตยกรรมแบบนี้เรียกว่า systolic arrays สำหรับการเอ็กซคิวแบบไปป์ไลน์ของอัลกอลิทึมที่เฉพาะเจาะจง

จากรูปแบบของระบบเครื่องคอมพิวเตอร์ทั้ง 5 แบบนี้ คอมพิวเตอร์แบบขนานที่ถูกสร้างขึ้นจะใช้รูปแบบ MIMD สำหรับใช้ในการคำนวณแบบทั่วๆไป โดยรูปแบบ SIMD และ MISD เหมาะสมสำหรับการคำนวณแบบที่มีจุดประสงค์พิเศษ ด้วยเหตุนี้เอง MIMD จึงเป็นรูปแบบที่มีการใช้กันอย่างแพร่หลายมากที่สุด SIMD รองลงมาและ MISD เป็นแบบที่ใช้กันน้อยที่สุด

2.4.4 คอมพิวเตอร์แบบขนาน และ เวคเตอร์คอมพิวเตอร์

คอมพิวเตอร์แบบขนานตามปกติแล้ว จะประมวลผลโปรแกรมในรูปของ MIMD เราสามารถแบ่งคอมพิวเตอร์แบบขนานได้เป็น 2 แบบหลัก ๆ คือ

1. Shared – memory multiprocessors
2. Massage – passing multicomputers หรือ Distributed – memory multicomputers

ความแตกต่างของ 2 แบบนี้จะอยู่ที่การจัดสรรหน่วยความจำและวิธีการที่ใช้ในการติดต่อสื่อสารระหว่างโพรเซสเซอร์ โพรเซสเซอร์ในระบบมัลติโพรเซสเซอร์จะติดต่อสื่อสารกับโพรเซสเซอร์ตัวอื่น ๆ ผ่าน shared variable ในหน่วยความจำที่ใช้ร่วมกัน แต่ละโหนดในระบบมัลติคอมพิวเตอร์จะมีหน่วยความจำเป็นของตัวเองซึ่งจะไม่แบ่งปันกันใช้กับโหนดอื่นๆ การติดต่อกันระหว่างโหนดจะทำในรูปแบบของเมสเซส (message passing) ที่เชื่อมต่ออยู่ระหว่างโหนด

บทที่ 3

Beowulf

3.1 บทนำ

Beowulf cluster เป็นโปรเจ็กต์ที่มีแนวคิดเริ่มต้นจาก องค์การนาซ่า เกิดขึ้นเมื่อปี 1994 มีจุดมุ่งหมายเพื่อที่จะสร้างคอมพิวเตอร์ที่ " ดีกว่าเดิม เร็วกว่าเดิม และถูกกว่าเดิม" ในการทำงาน ทำให้ใช้เวลาในการทำงานน้อยลง โดยได้มีการออกแบบโครงสร้างขึ้นโดยนาซ่าเพื่อที่จะนำมาใช้ทำงานที่ต้องใช้งานคิดคำนวณระดับสูง จำพวกงานด้านวิทยาศาสตร์, วิศวกรรมศาสตร์, การจำลองการบิน รวมถึงการควบคุมอุปกรณ์ที่ต้องการความละเอียดในเนื้องานสูง การประมวลผลสัญญาณ และการจำลองการทำงานต่างๆ

ระบบคอมพิวเตอร์ระดับสูง หรือที่เรียกว่า ซูเปอร์คอมพิวเตอร์ ที่มีความสามารถเฉพาะทางในการนำมาประมวลผลนั้น มีข้อจำกัดในการนำมาใช้ในหลายๆทาง ทั้งทางด้านราคาที่ค่อนข้างสูงมาก มีผู้ผลิตค่อนข้างน้อย ไม่ค่อยมีการร่วมกันพัฒนาให้เป็นไปในแนวเดียวกันระหว่างผู้ขาย ทำให้ไม่สามารถทำงานโปรแกรมเดียวกันระหว่างเครื่องต่างค่ายกันได้ และปัญหาที่สำคัญที่สุดคือ ด้านการขยายขีดความสามารถ โดยทั่วไปเครื่องระบบใหญ่ที่แม้จะมีความสามารถค่อนข้างสูง แต่ก็มีขีดความสามารถสูงสุดที่สามารถเป็นได้เช่นกัน ทำให้เมื่อเวลาผ่านไป ความต้องการในการคิดประมวลผลไม่พอเพียงต่อความต้องการ ทางเลือกใหม่ที่พยายามจะนำมาใช้คือ นำเครื่องที่ใช้งานทั่วไปที่ใช้เทคโนโลยีที่ใช้ในเครื่องเวิร์คสเตชันทั่วไปมาทดแทนเครื่องเดิม "Beowulf Cluster" นั่นเป็นโปรเจ็กต์ที่ทำการศึกษาและพัฒนา ถึงความเป็นไปได้ที่จะนำมาใช้ ในงานของนาซ่า ที่ต้องการการประมวลผลระดับสูง

3.2 โครงสร้างและประสิทธิภาพในการทำงาน

โครงสร้างของระบบ Beowulf เป็นการรวมกันของทั้ง ฮาร์ดแวร์ ซอฟต์แวร์ และการทำงานร่วมกัน ทำให้มีประสิทธิภาพที่ดีขึ้นต่อราคาที่เป็น โดยการทำงานต้องทำงานบนระบบเครือข่ายที่มีประสิทธิภาพ เพื่อเพิ่มความสามารถในการกระจายงาน ระบบ Beowulf Cluster เป็นระบบคลัสเตอร์ที่มีส่วนประกอบเป็น เครื่องพีซี โดยมีซีพียู โดยส่วนใหญ่จะมีซีพียูเป็น ตระกูล X86 ของ Intel เช่น 80486 Pentium PentiumII แต่จะไม่ใช้เครื่องสถาปัตยกรรมอื่นเช่น DEC Alpha หรือ Power PC Beowulf Cluster จะใช้ยูนิกส์เป็นระบบจัดการ (Operating System) เนื่องจากใช้ค่าใช้จ่ายค่อยข้างน้อย หรือ อาจจะไม่เสียเลย ค่าใช้จ่ายมีความสำคัญ เนื่องจากการเสียค่าใช้จ่ายในการซื้อระบบจัดการมาติดตั้งในแต่ละโหนดของระบบคลัสเตอร์ทำให้ผิดต่อจุดประสงค์หลักของ Beowulf Cluster ที่ต้องระบบราคาถูก การได้โค้ดของระบบจัดการก็สำคัญเช่นกัน เพราะจะทำให้สามารถแก้ไขปรับปรุงระบบให้มีความเหมาะสมต่อการทำงานแบบขนาน โดยทั้ง Linux และ BSD เป็นทางเลือกที่ดีในการนำมาใช้ติดตั้งในระบบ อย่างไรก็ตาม Solaris ก็สามารถนำมาใช้ในระบบเช่นกัน โดยมีผู้ใช้ไม่น้อยที่มั่นใจในการทำงานของ Solaris และสามารถทำงานได้อย่างดีเช่นกัน

ระบบ Beowulf นั้นมีการนำรูปแบบการทำงานแบบ Message Passing เข้ามาใช้ทำงาน โดยทั่วไปเราจะใช้การทำงาน เช่น MPI, PVM รวมถึง MPI-2 ทำงานรวมกับการทำงานแบบพื้นฐานโดยการใช้ sockets การทำงานโดยทั่วไป ผู้เขียนโปรแกรมจะต้องเขียนโปรแกรมจัดการ การใช้งานหน่วยความจำให้สามารถใช้งานหน่วยความจำของแต่ละเครื่องร่วมกันได้จึงจะทำให้ระบบมีประสิทธิภาพ การประมวลผลข้อมูลนั้นมีความสำคัญเป็นอย่างมาก โดยจุดหลักของ Beowulf คือการประมวลผลทางวิทยาศาสตร์และทางวิศวกรรม โดยการทำงานดังกล่าวมักมีทศนิยมเข้ามาเกี่ยวข้องเป็นอย่างมาก ซึ่งการประมวลผลดังกล่าวจะมีผลต่อการจัดการ การอ้างอิงหน่วยความจำ

Beowulf เป็นระบบหนึ่งที่ได้รับการแนะนำของระบบประมวลผลแบบขนานแบบต่างๆ ว่าเป็นระบบตัวอย่างของการทำงานแบบขนานที่ใช้อยู่ทั่วไป การทำงานของระบบคลัสเตอร์จะคิดว่าในแต่ละเครื่องโหนดมีระบบจัดการเป็นของตัวเอง โดยแต่ละโหนดจะติดต่อกัน ผ่านระบบเครือข่าย (LAN) ที่มีประสิทธิภาพสูง

โครงสร้างของ Beowulf ที่นาซ่าได้ทำการพัฒนา นั้นได้เป็นพื้นฐาน ทำให้มีการนำไปศึกษาต่ออย่างมากมาย โดยโปรเจ็กต์ Beowulf ได้เริ่มขึ้นอย่างจริงจังในปี 1993 โดยเป็นส่วนหนึ่งของโครงการสำรวจโลกและอวกาศของนาซ่า ที่ศูนย์การบิน Goddard Space โดยเรียกว่าการวิจัยให้ได้มาซึ่งเครื่องเวิร์กสเตชันที่มีความสามารถระดับ gigaflops โดยในขณะนั้นเครื่องที่มีความสามารถระดับนี้มีราคาประมาณ $50,000 ซึ่งถ้าใช้ระบบคลัสเตอร์ที่ทำจากครื่องพีซีราคาถูก ในโครงสร้างที่หน่วยความจุของฮาร์ทดิสก์เป็น 8 เท่าของความเร็วในการส่งข้อมูลของบัสระบบ(System Bus) โดยได้เครื่อง 16 เครื่องที่เป็น 80486 (100 MHz) ที่มีหน่วยความจำ 32 เมกกะไบต์ และมีความจุของฮาร์ดดิสก์เป็น 1.6 Gbyte ในแต่ละเครื่อง โดยแต่ละเครื่องต่อเข้าด้วยกันด้วย อีเทอร์เน็ทที่ความเร็ว 10 Mbps จะทำให้ระบบมีความสามารถประมาณเทียบเท่า 1 gigaOPS โดยในแต่ละเครื่องจะต้องมีระบบจัดการที่เหมือนกับที่ใช้งานอยู่ มีการติดตั้งโปรแกรมตามที่ต้องใช้โดย ในระบบตัวอย่างจะใช้ลีนุกส์เป็นระบบปฏิบัติการ โดยการทำงานจะมีข้อที่ทำให้ความสามารถของระบบต่ำลงที่เด่นชัดจะเป็นส่วนของโปรแกรมจัดการทางด้านเน็ตเวิร์กเป็นจุดสำคัญที่ต้องปรับปรุงให้มีประสิทธิภาพดีขึ้น ทำให้เกิดโครงการต่อ ๆ มาที่จะสร้างการติดต่อที่มีประสิทธิภาพดียิ่ง ๆ ขึ้น

3.3 เทคโนโลยีทางด้านเครือข่ายและการขยายเครือข่าย

การติดต่อสื่อสารผ่านเครือข่าย ที่ต่อเครื่องแต่ละโหนดเข้าด้วยกัน ต้องมีความสามารถพอที่จะตอบสนองความต้องการของโปรแกรมที่ทำงานอยู่และต้องมีราคาที่ไม่แพงจนเกินไปต่อประสิทธิภาพที่ได้มา เราสามารถคาดเดาได้ว่า การเพิ่มขึ้นของจำนวนโหนดนั้นจะมีผลทำให้ความต้องการในการใช้เครือข่ายเพิ่มขึ้นเป็นลำดับ โดยความสำคัญของขีดความสามารถของเครือข่ายจะมีผลต่อความสมดุลย์ของการประมวลผลข้อมูลโดยจุดมุ่งหมายแล้ว จะต้องทำให้ ระบบเครือข่ายไม่เป็นคอขวดต่อการประมวลผล ซึ่งโดยทั่วไปแล้ว ราคาของระบบเครือข่ายจะประมาณ 1 ใน 4 ส่วนของราคาทั้งระบบ

ทางเลือกใหม่ในระบบเครือข่ายของระบบคลัสเตอร์ คือระบบเครือข่ายแบบฟาสอีเทอร์เน็ต สามารถส่งข้อมูลที่ความเร็วสูงสุดถึง 100 Mbps โดยแต่ละโหนดจะติดตั้ง เน็ตเวิร์กการ์ด ซึ่งในแต่ละโหนดจะเสียค่าใช้จ่ายประมาณ $50 โดยระบบนี้สามารถส่งข้อมูลระหว่างโหนด ที่ความเร็ว 100 μs หรือบางครั้งน้อยกว่า 80 μs

อีกทางเลือกหนึ่งที่ของระบบเครือข่ายความเร็วสูงที่เหมาะที่จะนำมาใช้ในระบบคือ Myrinet เป็นระบบที่พัฒนามาเพื่อการใช้งานระหว่างเครื่องในเครือข่ายขนาดเล็ก หรือที่เรียกว่า System Area Network (SAN) โดยความสามารถส่งข้อมูลได้ 1 Gbps และมีความเร็วในการเข้าถึงที่ 20 μs โดยต่อเชื่อมเข้ากับสวิตซ์ 8 พอร์ต โดยราคาต่อ โหนดประมาณ $3,000 ซึ่งค่อนข้างแพงแต่เมื่อเทียบกับประสิทธิภาพที่ได้แล้วก็คุ้มสำหรับงานที่ต้องใช้ความสมารถของระบบเครือข่ายสูง

ระบบคลัสเตอร์ขนาดเล็ก 16 ถึง 24 โหนด ที่ต่อเข้ากับฟาสอีเทอร์เน็ตสวิตซ์ตัวเดียว เป็นระบบที่เป็นพื้นฐาน และมีประสิทธิภาพค่อนข้างดีเมื่อเทียบกับราคา

อีกระบบที่จะกล่าวถึงเป็นระบบที่ถูกสร้างขึ้น ให้มีความสามารถในการส่งข้อมูลสูงเพื่อเป็นเส้นทางหลักในการส่งข้อมูลของระบบทั้งหมด ตัวอย่างเช่น Prominet P550 เป็นสวิตซ์ที่สามารส่งข้อมูลที่ความเร็ว 22 Gbps สามารถต่อเข้ากับ ฟาสอีเทอร์เน็ต 20 พอร์ต และสามารถเพิ่มได้มากที่สุดที่ 120 พอร์ต ทำให้สามารถติดต่อเข้ากับโหนดจำนวนมาก

3.4 ซอฟต์แวร์ที่ใช้ในระบบรวมถึงโปรแกรมที่ช่วยในการทำงาน

ระบบ Beowulf นั้นต้องการระบบที่จะทำงานร่วมด้วย ที่มีประสิทธิภาพสูงและมีเสถียรภาพค่อนข้างดี เพื่อใช้ทำงานรวมกับโปรแกรม และการการจัดการส่วนต่างๆของระบบ ส่วนต่างๆที่สำคัญต่อระบบเป็นดังนี้

- Application Development
- Debugging/Tuning Tools
- Low-level Programming Interface
- Operation System Services

หัวข้อเหล่านี้คือจุดมุ่งหมายในการที่นาซ่าได้จัดเวิร์คช็อปขึ้นมา เพื่อศึกษาความเป็นไปได้ในการสร้างและพัฒนาโปรแกรมขึ้น และคาดการณ์ถึงความต้องการของระบบคลัสเตอร์ในอนาคต โดยโปรแกรมและทูลต่างๆ ที่มีอยู่แล้วและรวมถึงที่ยังอยู่ระหว่างการพัฒนา มีดังนี้

Software/Tool Areas	Name/Status

Application Development

Utilities Language	Sh , csh , grep, egrep, sed, diff, vi, ex, make, etc; all avilable via GNU
	F77, C, C++, f90, linker, mixed language support
	GNU C, C++, f77
	PGI C, C++, f77, HPF
	Absoft f77, f90
	NAG f77, f90

Debugging/Tuning Tools

Interatctive Parallel Debugger/Postmortem Debugger	ARC P2D2 project is being ported to beowulf-class clusters
Profiling Tools	Single Processor; Standard Gprof Utility
Event/Message-Passing Tracing Tools	PABLO Project ARC AIMS project may be ported to Beowulf-class cluster
Hardware Performance Mornitoring Tools	LANL has the beginnings of a hardware monitoring tool

Low-level Programming Interface (Libraries)

Messapge Passing	MPI, PVM Argonne MPICH, LAM MPI, publicly available PVM
POSIX Treads	Available
Math Libraries(Optimized Single Processor)	FFT,BLAS,Random; BLAS and Random Number Generator Available
Math Libraries (Parallel)	FFT, Plapack Available Array Permutation

Software/Tool Areas	Name/Status
Parallel I/O	Timer (Wall-Clock, System,and User Time) All Standard Features Of linux Clemson PVFS parallel file system ARC PMPIO MPI-2 I/O library (available) Argonne Romio MPI-2 I/O library ARC ESP FS parallel file system

Operating System Service

TCP/IP	Standart Part of Linux
File System (w/Long Names, >4GB File Systems, >4GB Files)	Available, Except > 4GB Files Which is in Development
Job Queuing and Scheduling (Batch, Space-sharing, Time-sharing)	Easy LoBoS Queueing System ARC PVS Project
Accounting	ARC Acct++ Site-wide Accounting Project (will be ported to Linux)
Quotas and Limits Enforcement	

ตารางที่ 3.1 แสดง tools ต่างๆ ที่กำลังอยู่ระหว่างการพัฒนา

3.5 การขยายขนาดของระบบ

Beowulf ในยุคแรกจะมีขนาดเล็กถึงกลาง โดยมีขนาด 8, 16 หรือ 32 โหนด และมีอย่างมากที่สุด 48 โหนด โดยโครงสร้างที่มีขนาดนี้ ทำให้ได้ประสิทธิภาพไม่มากนัก เพียงไม่กี่ Gflops ในการประมวลผล การเพิ่มขนาดโครงสร้าง ขยายให้มีประสิทธิภาพสูงขึ้นตามความต้องการ

การวัดข้อมูลต่าง ๆ

การขยายขนาดของระบบ ซึ่งจะมีความสัมพันธ์กับค่าต่างๆของระบบโดยรวมได้แก่ค่าดังต่อไปนี้

- flops
- Ops
- ขนาดของหน่วยความจำ
- ขนาดของดิสก์ที่เหลือ
- อัตราการเข้าใช้ดิสก์
- ประสิทธิภาพของเน็ตเวิร์ก
- อัตราการส่งถ่ายข้อมูลในเน็ตเวิร์ก

ประสิทธิภาพ

ประสิทธิภาพนั้นเป็นสัดส่วน และส่งผลต่อคุณสมบัติของระบบที่ประมวลผลแบบหลายซีพียู ประสิทธิภาพ เป็นส่วนหนึ่งของปริมาณงานและขึ้นกับขนาดของระบบ ถ้ามองอย่างง่าย ๆ ก็คือเมื่อเราเพิ่มจำนวนโหนด อีกทางเลือกในการเพิ่มประสิทธิภาพคือ ลดปัญหาต่าง ๆ เช่นปัญหาทางเน็ตเวิร์กหรือทางอื่น เพื่อทำให้ใช้เวลาในการทำงานลดลงและส่วนนั้นจะกลายเป็นประสิทธิภาพที่เพิ่มขึ้น

ราคาต่อโหนด

ราคาต่อโหนด มีความสำคัญต่อการทำระบบ Beowulf ดังนั้นการคำนึงถึงราคาต่อโหนด รวมถึงประสิทธิภาพที่จะคุ้มค่าต่อราคาที่จะเสียไปต่อโหนดหรือไม่ เราต้องคำนึงถึงว่าถ้าเราซื้อเครื่องที่มีประสิทธิภาพด้อยกว่าและมีราคาที่ถูกกว่าก็จริง แต่เราจะต้องเสียค่าใช้จ่ายเพิ่มขึ้นไปกับค่าติดตั้งเครือข่ายไปเท่ากับจำนวนโหนดที่เพิ่มขึ้นอย่างเป็นสัดส่วน

จำนวนผู้ใช้และโปรแกรมที่ใช้

จำนวนผู้ใช้และโปรแกรมที่สามารถทำได้ในช่วงเวลาหนึ่ง ๆ มีความสำคัญในระบบที่มีความต้องทำงานประเภทนี้ โดยระบบที่มีขนาดเล็กและกลาง ซึ่งประมาณมูลค่าของระบบต่ำกว่า S100K ส่วนมากจะนำมาใช้ในงานส่วนตัว สามารถทำงานได้ทีละ 1 คน แต่หากเป็นระบบขนาดใหญ่ซึ่งระบบอาจจะมีมูลค่ากว่า S0.5M ซึ่งสามารถนำมาใช้หลายคนได้โดยที่ประสิทธิภาพไม่ตก และยังทำให้ใช้ระบบได้คุ้มค่า

3.6 การเกิดคอขวดทางฮาร์ดแวร์

การทำงานบนระบบคลัสเตอร์ มีงานหลายแบบด้วยกันอย่างเช่น การประมวลผลฟูเรียร์ เป็นการประมวลผลทางคณิตศาสตร์ระดับสูง ที่ใช้การทำงานของ ซีพียูและข้อมูลในเครื่องมากกว่าการใช้งานเน็ตเวิร์ก ทำให้มีการใช้งานของเน็ตเวิร์กน้อย แต่การที่ใช้ซีพียูและข้อมูลในเครื่องมาก อาจจะทำให้เกิดปัญหาคอขวดขึ้นในเครื่องได้ เช่น มีการใช้งานฮาร์ทดิสก์มาก ๆ ก็อาจทำให้บัสที่ติดต่อกับฮาร์ทดิสก์ เกิดคอขวดเป็นปัญหาได้

มีระบบเน็ตเวิร์กรูปแบบไม่มากที่สนับสนุนการทำงานของ Beowulf Cluster จำนวนมาก ๆ Myrinet สวิตซ์มีขนาด 8 พอร์ต และมีบล็อก หากต้องการขยายขนาดของเครือข่ายสามารถขยายโดยผ่านทางบล็อกที่มีมาได้ ทำให้สามารถทำงานได้อย่างดีถึงแม้จะมีการขยายขนาดของการทำงานก็ตาม หรืออีกทางคือ ฟาสอีเทอร์เน็ต ที่มีผู้ขายออกแบบระบบเครือข่ายจาก สวิตซ์ (4,16 หรือ 24 พอร์ต) โดยใช้การเร้าระหว่างสวิตซ์ด้วยอัลกอริทึมแบบทรี แต่การออกแบบแบบนี้ก็ยังไม่สามารถแก้ปัญหาคอขวดของระบบเครือข่าย จึงมีการออกแบบการออกแบบ สวิตซ์ที่มีความเร็วสูงเพื่อนำมาเป็นตัวหลักของสวิตซ์ตัวลูกที่มีความเร็วด้อยกว่า ทำให้ลดปัญหาคอขวดได้ทำให้ประสิทธิภาพสูงขึ้น

ฟาสอีเทอร์เน็ต เป็นทางเลือกที่ดีที่จะนำมาใช้ในระบบขนาดเล็ก ส่วน Myrinet นั้นเร็วทำให้สามารถสร้างเป็นระบบใหญ่ ๆ ได้โดยไม่เกิดปัญหา แต่ค่าใช้จ่ายในการเริ่มต้นค่อนข้างสูง จึงมีการนำ Myrinet มาผสมผสานกับฟาสอีเทอร์เน็ต โดยให้มี Myrinet เป็นส่วนกลางของระบบที่ต้องรับโหลดสูง และมีฟาสอีเทอร์เนทเป็น ส่วนทรี ทำให้ได้ประสิทธิภาพที่ดีขึ้น

ความสามารถในการส่งของข้อมูลนั้นสามารถูกกำหนดได้ ไม่เฉพาะที่เครือข่าย อาจจะถูกกำหนดด้วยบัสของอินเตอร์เฟซก็ได้ ปรกติอาจจะเป็น 32-bit PCI โดยความสามารถของบอร์ดใหม่ๆอาจจะทำให้ได้ประสิทธิภาพเพิ่มขึ้นเป็น 64-bit PCI ได้ ทำให้สามารถทำงานได้รวดเร็วขึ้น ลดคอขวด

การจำกัดแบนด์วิดธ์ของหน่วยความจำ มีผลกระทบต่อประสิทธิภาพของการทำงานของระบบจำนวนของซีพียู (ในโหนดเดียว) ซึ่งปัจจุบันสามารถมีได้ถึง 8 ซีพียูใน 1 เครื่อง การที่มีซีพูยูมาก อาจทำให้แบนด์วิดธ์ของหน่วยความจำมีไม่เพียงพอ ที่จะสนับสนุนให้ซีพียูเหล่านี้ทำงานได้อย่างเต็มที่ยกเว้นแต่มีการเขียนโปรแกรมให้จัดการแคชให้ใช้งานได้ประสิทธิภาพ

การมีปัญหาในระดับฮาร์ดแวร์ก็เป็นปัญหาหนึ่งที่ทำให้ประสิทธิภาพการทำงานลดลง เนื่องจากการทำงานต้องการความต่อเนื่องเป็นอย่างมาก อาจเป็นเดือนๆที่ไม่ได้ปิดเครื่อง การที่เครื่องมีปัญหาต่างๆ เช่น ดิสก์, พัดลมซีพียู, เพาเวอร์ซัพพลาย และอื่นๆ การขจัดปัญหาเหล่านี้จะทำให้การทำงานของระบบเป็นไปได้อย่างต่อเนื่อง

3.7 การเกิดคอขวดทางซอฟต์แวร์

การเพิ่มขนาดการทำงานตามสัดส่วน จะต้องคำนึงถึงทั้งประสิทธิภาพและความสามารถในการใช้งาน เมื่อมีการเพิ่มจำนวนโหนด ในการพิจารณาด้านประสิทธิภาพ โปรแกรมจะต้องมีการเขียนให้สนับสนุนการทำงานแบบขนาน และปรับแต่งให้มีประสิทธิภาพสูงสุดเท่าที่จะเป็นไปได้ที่จะเกิดขึ้นในการแบ่งงาน และการใช้งานโปรแกรมนั้นในระบบที่ใหญ่ขึ้นจะต้องมีการจำลอง เอ็นไวรอนเมนต์ (environment) ให้เหมือนเครื่องที่ทำงาน

การจำลองเอ็นไวรอนเมนต์เป็นความต้องการของโปรแกรมในการทำงาน เพื่อให้สามารถทำงานได้อย่างถูกต้องทั้งชื่อทั้ง พาร์ท รวมถึงค่าตัวแปรระบบต่าง ๆ รวมถึงคำสั่งต่างๆที่จำเป็นต่อการทำงาน ซึ่งควรมีอยู่ในระบบ เช่นคำสั่ง killall

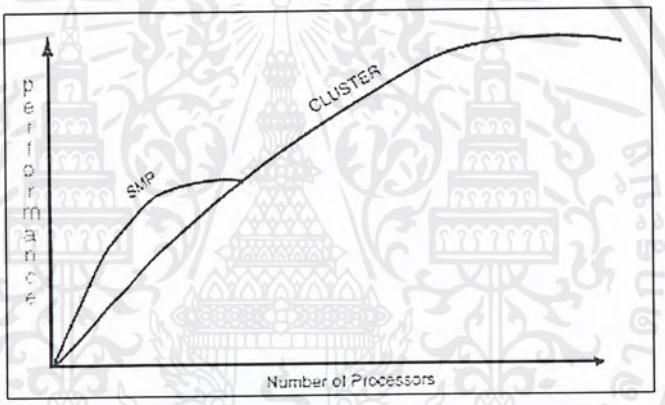
จะมีประสิทธิภาพด้อยกว่าการเข้าถึงเครื่องของตัวเอง โดยประสิทธิภาพที่ด้อยลงไปนี้อาจมาจากสาเหตุอื่น ๆ เช่น โพรโตคอล

ในทางตรงกันข้าม ระบบ SMPs จะเป็นระบบที่มีตัวประมวลผลหลายตัว แต่สำหรับอินพุต/เอาต์พุต และหน่วยความจำจะมีเพียงอย่างละ 1 ตัว โดยตัวประมวลผลทุกตัวจะมีประสิทธิภาพที่เท่ากัน

ถึงแม้ว่าการเปรียบเทียบส่วนใหญ่จะถูกครอบคลุมโดยเนื้อหาข้างต้น ก็ยังมีตัวแปรที่เกี่ยวกับโครงสร้างทางฮาร์ดแวร์ของคลัสเตอร์ซึ่งเป็นสาเหตุให้มีตัวแปรเพิ่มขึ้นในรายละเอียดของการพิจารณาในเนื้อหาต่อไปนี้

3.8.2 การพิจารณาในรายละเอียด

ประสิทธิภาพไม่ใช่หัวข้อเดียวที่จะนำมาเปรียบเทียบระหว่างคลัสเตอร์กับระบบ SMPs โดยจากกราฟเราสามารถสังเกตได้เพียงแต่ว่า เมื่อมีการเพิ่มจำนวนของตัวประมวลผลให้กับระบบมากขึ้นจะทำให้กราฟมีการโค้งลงจากจุดสูงสุดของกราฟอย่างเห็นได้ชัด ซึ่งจะเป็นในทั้งคลัสเตอร์และระบบ SMPs



รูปที่ 3.1 แสดงการเปรียบเทียบประสิทธิภาพเมื่อจำนวนโปรเซสเซอร์มีจำนวนเพิ่มขึ้น

จากกราฟในแนวตั้งแสดงให้เห็นว่าระบบที่มีจำนวนของตัวประมวลผลมากๆ ระบบ SMPs จะเกิดปัญหากับ แบนด์วิดช์ของหน่วยความจำ, แบนด์วิดช์ของอินพุต/เอาต์พุต สำหรับคลัสเตอร์จะไม่ประสบปัญหานี้เพราะว่ามีอุปกรณ์เสริมเป็นจำนวนมากสำหรับการสื่อสารระหว่างโหนดแต่จะมีผลทำให้คลัสเตอร์ทำงานได้ยากในการทำกระจายงาน , การจัดการคิว และ การติดต่อระหว่างโหนดโดยอินพุต/เอาต์พุต

สำหรับ SMPs จะใช้ไฟฟ้าในการทำงานสูงและราคาของการจัดหาหน่วยความจำและแบนด์วิดช์ของอินพุต/เอาต์พุตที่เพียงพอจะค่อนข้างสูง ซึ่งจะทำให้เกิดการแย่งกันใช้งานอุปกรณ์ในระบบ

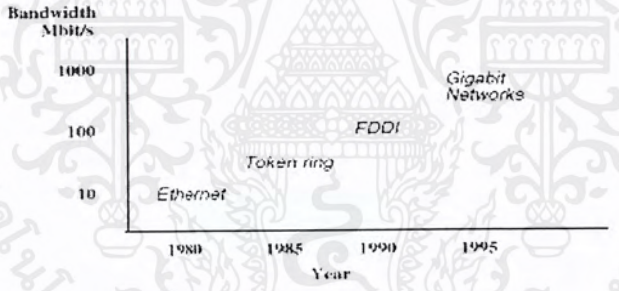
เนื่องจากคลัสเตอร์มีความสมดุลในเรื่องของหน่วยความจำและความสามารถของอินพุต/เอาต์พุต เมื่อคุณทำการเพิ่มโหนดให้กับระบบคุณก็จะเพิ่ม หน่วยความจำ และ อินพุต/เอาต์พุต ให้กับระบบด้วย ซึ่งเป็นการรักษาสมดุลของประสิทธิภาพของระบบไปเรื่อยๆ เหนือกว่าที่ระบบ SMPs จะทำได้โดยระบบคลัสเตอร์จะไม่ใช้ตัวประมวลผลจำนวนน้อย ๆ เพราะจะทำให้ประสิทธิภาพของระบบต่ำ

ถ้าจะให้ระบบมีประสิทธิภาพที่ดีมากกว่าคลัสเตอร์ธรรมดา ในแต่ละโหนดของคลัสเตอร์ควรเป็น SMPs ซึ่งจะทำให้ระบบมีประสิทธิภาพดีกว่าการใช้ SMPs เพียงเครื่องเดียว มีหลายโปรแกรมประยุกต์ที่สามารถทำงานได้บนระบบ SMPs แต่ไม่สามารถทำได้กับคลัสเตอร์เพราะว่าไม่สามารถแสดงข้อมูลขนาดใหญ่ได้ เนื่องมาจากอุปกรณ์เสริมในการติดต่อสื่อสารระหว่างเครื่อง แบนด์วิดช์มาตรฐานของการสื่อสารแบบเปิดสำหรับระบบคลัสเตอร์ที่มี N เครื่องสามารถขยายใหญ่ได้มากกว่าบัสภายในของระบบ SMPs ที่มีจำนวน N ตัวประมวลผล

3.9 เทคโนโลยีการเชื่อมต่อแบบและการพัฒนาในรูปแบบต่างๆ

ตามปกติ เครื่องเวิร์กสเตชั่นเพียงเครื่องเดียวสามารถทำงานได้ถึง 10 ล้านคำสั่งใน 1 วินาที แต่เรายังสามารถเพิ่มประสิทธิภาพได้โดยการเชื่อมต่อระบบเข้ากับ high speed network ซึ่งจะทำให้สามารถประมวลผลโปรแกรมประยุกต์ต่างๆที่ซับซ้อนได้ อาจจะมีความสามารถถึงระดับเดียวกับซูเปอร์คอมพิวเตอร์

การจะใช้การประมวลผลแบบ distributed computing ได้มีประสิทธิภาพดีนั้นมีองค์ประกอบหลักอยู่ที่ความเร็วในการเชื่อมโยงข้อมูล



รูปที่ 3.2 วิวัฒนาการของ Networking speed

- **Ethernet** เป็นรูปแบบการเชื่อมต่อซึ่งใช้การส่งข้อมูลแบบ packet switching มีความเร็วในการส่งแบบ Broadcast เท่ากับ 10 Mbit/sec โดยใช้ bus ติดต่อระหว่างเครื่องคอมพิวเตอร์
- **FDDI** เป็นการเชื่อมต่อแบบที่ใช้ optical fiber เป็นตัวกลาง มีความเร็วในการส่งข้อมูลเท่ากับ 100 Mbit/sec
- **HiPPI** เป็นการเชื่อมต่อแบบขนานที่มีประสิทธิภาพในการส่งข้อมูลสูง โดยใช้สายทองแดงหลายๆเส้นเป็นตัวกลางในการส่งข้อมูล ความเร็วในการส่งข้อมูลเท่ากับ 800 Mbit/sec สำหรับแบบ 32 เส้น และความเร็วเท่ากับ 1.6 Gbit/sec สำหรับแบบ 64 เส้น

- **SONET**(Synchronous Optical Network) เป็นวิธีการในการส่งข้อมูลแบบอนุกรมซึ่งมีหลายอัตราการส่ง ได้แก่ OC-1 มีอัตราเร็ว 51.84 Mbit/sec, OC-3 มีอัตราเร็ว 155.52 Mbit/sec, OC-12 มีอัตราเร็ว 622.08 Mbit/sec และ OC-192 มีอัตราเร็ว Gbit/sec
- **ATM** (Asynchronous Transfer Mode) เป็นเทคนิคในการส่งข้อมูล การทำ multiplexing และการ switching ซึ่งจะมีความเหมาะสมในการส่งข้อมูลเพิ่มขึ้น โดยได้กำหนดขนาดของ packet ดังนี้ คือ header มีขนาด 5 byte และส่วนของข้อมูลมีขนาด 48 byte

บทที่ 4
ระบบคลัสเตอร์ที่ได้ทำการออกแบบ

4.1 บทนำ

ระบบคลัสเตอร์ที่ทำการออกแบบ เน้นตามจุดประสงค์เดิมของ Beowulf cluster ที่มีจุดมุ่งหมาย 3 ข้อคือ " ดีกว่าเดิม, เร็วกว่าเดิม และถูกกว่าเดิม" แต่ส่วนที่สำคัญที่สุดก็คือ ความประหยัด ในหัวข้อ โครงงานที่มีชื่อว่า Poor's Man SuperComputer มีจุดประสงค์หลักในการทดสอบระบบคลัสเตอร์บนทรัพยากรที่มีอยู่จำกัด ทั้งจำนวน และ ประสิทธิภาพ รวมถึงศึกษาเพื่อหาแนวทางการใช้งานของระบบ คลัสเตอร์ในรูปแบบต่างๆ เพื่อนำมาสร้างระบบตัวอย่างในการทดสอบประสิทธิภาพ อีกจุดประสงค์ที่มีความสำคัญไม่ยิ่งหย่อนต่อกัน ก็คือ การศึกษาเพื่อรวบรวมการทำงาน การติดตั้งแบบต่างๆ เพื่อเป็นแนวทางในการนำมาประยุกต์ใช้ในการสร้างระบบในอนาคต เพื่อใช้ทดแทนระบบซูเปอร์คอมพิวเตอร์ที่ใช้งานอยู่ในปัจจุบันได้อย่างคุ้มค่า

โครงสร้างของระบบคลัสเตอร์ทั่วไปที่ได้กล่าวมาแล้วในข้างต้นเป็นการนำเครื่องเวิร์กสเตชันที่มีความสามารถระดับหนึ่ง มีหน่วยความจำและฮาร์ทดิสก์ที่พอเหมาะนำมารวมเข้าเป็นระบบขนาดเล็กถึงขนาดกลาง ทำให้ได้ประสิทธิภาพที่ดีในระดับที่น่าพอใจ โดยทั่วไปในระบบเริ่มแรกที่ได้ทดสอบของ Beowulf คลัสเตอร์ได้นำ เครื่อง 486DX ที่ทำงานที่ 100 Mhz มีหน่วยความจำ 16 เมกะไบต์ มีฮาร์ทดิสก์ที่ติดตั้งระบบปฏิบัติการลีนุกส์ ทั้งหมด 16 เครื่อง โดยทั้งหมดเชื่อมต่อกันโดยผ่านทางสวิตซ์ ระบบดังกล่าวเป็นระบบที่ได้ออกแบบมาให้มีประสิทธิภาพสูงเมื่อเทียบกับระบบคลัสเตอร์ที่มีอยู่ในขณะนั้น

การออกแบบระบบคลัสเตอร์ที่ได้คิดขึ้น ได้พยายามลดค่าใช้จ่ายในส่วนต่าง ๆ ลง ส่วนแรกที่ปรับเปลี่ยนจากเดิมคือ ตัดส่วนของฮาร์ดดิสก์ออกจากระบบ โดยนำระบบ BOOTP เข้ามาทดแทน ส่วนของฮาร์ทดิสก์ที่ได้ตัดไป โดยให้เครื่องโหนดทุกเครื่องบูตระบบผ่านทางระบบเครือข่ายไปยังเครื่องเซิร์ฟเวอร์ ทำให้เครื่องโหนดทำงานได้ปกติ เสมือนมีฮาร์ทดิสก์ในเครื่อง แล้วทำการติดตั้ง โปรแกรมช่วยต่าง ๆ ที่ช่วยในให้สามารถทำงานแบบขนานได้ โดยมีเครื่องกลางในการควบคุมให้การทำงานเป็นไปอย่างมีระบบระเบียบ โดยจะเรียกว่า "คลัสเตอร์ คอนโทรลเลอร์" รวมถึงปรับแต่งระบบให้มีความเหมาะสมต่อสภาพเครื่องที่นำมาติดตั้งด้วย

โครงสร้างของระบบที่ได้กล่าวมาข้างต้นมีส่วนประกอบหลักหลายส่วนด้วยกัน โดยที่จะกล่าวถึงมีหลายส่วนดังนี้

- การทำงานแบบดิสก์เลส
- การทำงานของ NIS
- การทำงานของ NFS
- โครงสร้างและการทำงานของ PVM

- โครงสร้างและการทำงานของ MP
- โปรแกรมโมสิคซ์

4.2 การทำงานของแบบดิสก์เลส

คำว่า ดิสก์เลส (**Diskless**) หมายความว่า ไม่มีดิสก์ ดังนั้นระบบการทำงานแบบดิสก์เลสจึงหมายความว่า ระบบที่ไม่จำเป็นต้องมีดิสก์ของตัวเองในการปฏิบัติงาน (การจัดเก็บข้อมูล, ระบบปฏิบัติการ เป็นต้น) โดยระบบการทำงานแบบนี้จะเก็บข้อมูลข่าวสารที่จำเป็นสำหรับการทำงาน รวมทั้งระบบปฏิบัติการ ไว้ในดิสก์ที่เครื่องเซิร์ฟเวอร์เพียงเดียวเท่านั้น แล้วเครื่องใดที่ต้องการข้อมูลนั้น ๆ ไปใช้งานก็จะส่งสัญญาณร้องขอไปยังเครื่องเซิร์ฟเวอร์ เมื่อเครื่องเซิร์ฟเวอร์ได้รับสัญญาณร้องขอแล้ว ก็จะจัดข้อมูลที่เครื่องที่ร้องขอต้องการส่งไปกลับไปให้ เป็นการทำงานในลักษณะไคลเอนท์ – เซิร์ฟเวอร์ ข้อดีของระบบการทำงานแบบดิสก์เลสนี้ คือ

1. ประหยัดค่าใช้จ่ายในเรื่องของดิสก์ การอัพเกรดซอฟท์แวร์
2. การสำรองข้อมูล (Backup) และการกู้คืน (Recovery) จะทำที่เครื่องเซิร์ฟเวอร์เครื่องเดียว ทำให้ง่ายต่อการดูแลและจัดการระบบ

<u>ขั้นตอนการบูตผ่านระบบเน็ตเวิร์ก</u>

การทำงานของระบบดิสก์เลสนี้จำเป็นที่จะต้องการบูตผ่านระบบเน็ตเวิร์ก เนื่องจากเครื่องไคลเอนท์แต่ละเครื่องจะไม่มีดิสก์เป็นของตัวเอง ทำให้ไม่มีการจัดเก็บข้อมูล และระบบปฏิบัติการในเครื่องตัวเอง และในการบูตผ่านระบบเน็ตเวิร์กนี้ เครื่องคอมพิวเตอร์แต่ละเครื่องจะต้องประกอบไปด้วย 1. สิ่งบ่งบอกถึงความเป็นตัวเอง 2. อิมเมจของระบบปฏิบัติการ และ 3. ไฟล์ซิสเต็มที่สามารถทำงานได้

มาพิจารณาเครื่องคอมพิวเตอร์ที่เป็นดิสก์เลส ในระบบเน็ตเวิร์กหนึ่ง ๆ อาจจะมีเครื่องคอมพิวเตอร์ที่เป็นดิสก์มากกว่าหนึ่งเครื่องก็ได้ แล้วทำอย่างไรจึงจะสามารถแยกเครื่องเหล่านั้นออกจากกันได้ ทำได้โดยมีข้อมูลข่าวสารชนิดหนึ่งซึ่งจะมีแค่หนึ่งเดียวเท่านั้นในเครื่องคอมพิวเตอร์แต่ละเครื่อง (การ์ดเน็ตเวิร์กของแต่ละเครื่อง) โดยการ์ดเน็ตเวิร์กทุก ๆ อันในโลกนี้จะมีตัวเลขเฉพาะตัวของแต่ละอันจำนวน 48 บิต เพราะว่าผู้ผลิตการ์ดเน็ตเวิร์กแต่ละรายนั้นจะได้รับค่าที่กำหนดไว้สำหรับการผลิตอยู่ก่อนแล้ว โดยตัวเลขแอดเดรสของการ์ดเน็ตเวิร์ก จะอยู่ในรูปเลขฐาน 16 และแบ่งโดยเครื่องหมายโคลอนเป็น 6 กลุ่ม ๆ ละ 2 ตัว ตัวอย่างเช่น 00:60:08:C7:A3:D8

โพรโตคอลที่ใช้ในการแจกจ่ายไอพีแอดเดรส โดยมีเน็ตเวิร์กการ์ดแอดเดรสอ้างอิง คือ Boot Protocol (BOOTP) และ Dynamic Host Configuration Protocol (DHCP) ในหัวข้อนี้เราจะเลือกใช้ BOOTP เนื่องจาก มีขนาดเล็กกว่า และ ก็มีความสามารถเพียงพอในการทำงานในหัวข้อนี้แล้ว

ตัวอย่างการทำงานของ BOOTP

ไคลเอนท์ : ส่งสัญญาณออกไปว่า เน็ตเวิร์กการ์ดแอดเดรสของฉัน คือ 00:60:08:C7:A3:D8 กรุณาส่งไอพีแอดเดรสของฉันมา

เซิร์ฟเวอร์ : (ค้นหาข้อมูลในฐานข้อมูล) ชื่อของคุณคือ mercury02 , ไอพีแอดเดรสคือ 161.246.6.72 , เซิร์ฟเวอร์คือ 161.246.6.71 , ไฟล์ที่ใช้ในการบูตคือ /tftpdir/vmlinuz.nfs

โดยในตอนแรกเครื่องไคลเอนท์จะไม่รู้ไอพีแอดเดรสของเครื่องเซิร์ฟเวอร์ แต่ไคลเอนท์จะทำการกระจายสัญญาณคำขอในระบบแลน (LAN) แล้วเครื่องเซิร์ฟเวอร์ที่ได้รับคำขอนั้นก็จะส่งข้อมูลตอบกลับไปให้เครื่องไคลเอนท์นั้น ๆ

หลังจากที่เครื่องไคลเอนท์ได้ไอพีแอดเดรสแล้ว ไคลเอนท์ก็จะดาวน์โหลดอิมเมจของระบบปฏิบัติการและ ปฏิบัติงานต่อไป ในขั้นตอนนี้จะมีอีกหนึ่งโพรโตคอลที่นำมาใช้งาน เรียกว่า Trivial File Transfer Protocol (TFTP) ซึ่ง tftp นี้เป็นเวอร์ชันต่ำๆ ของ ftp โดยจะไม่มีการรับรองสิทธิ (authentication) และการทำงานจะทำงานผ่าน User Datagram Protocol (UDP) แทน Transmission Control Protocol (TCP) เหตุผลที่ใช้ UDP แทนเพราะว่า โค้ดมีขนาดที่เล็กกว่าและง่ายในการส่งข้อมูลและตรวจสอบความผิดพลาดในการส่ง โดยจะส่งข้อมูลในลักษณะเป็นบล็อก ๆ

ขั้นตอนสุดท้ายในการทำงานระบบปฏิบัติการ รูทไฟล์ซิสเต็มจะถูกสร้างขึ้น และโพรโตคอลที่ใช้งานในขั้นตอนนี้คือ Network File System (NFS) ซึ่งจะกล่าวในหัวข้อต่อ ๆ ไป

การนำระบบดิสก์เลสมาประยุกต์ใช้ในระบบคลัสเตอร์ที่ได้กล่าวมานี้ ถึงแม้จะมีข้อดีในการลดค่าใช้จ่ายในการติดตั้ง และช่วยขจัดปัญหาการทำงานการดูแลจัดการข้อมูลก็ตาม แต่ก็มีข้อเสียที่ต้องคำนึงถึงเสมอ คือ การทำดิสก์เลส จะเป็นการดึงข้อมูล ทั้งหมดจากเซิร์ฟเวอร์ที่ให้บริการในเครือข่ายทำให้ทุกครั้งที่จะทำคำสั่งใด ๆ ในระบบ หรือ ต้องการดึงข้อมูลใด ๆ ในระบบ จะต้องสั่งผ่านระบบเครือข่าย เพื่อไปเรียกข้อมูลจากเซิร์ฟเวอร์ ทำให้สูญเสียแบนด์วิดธ์ ไปกับการทำงานประเภทนี้ ของทุก ๆ เครื่องที่ทำดิสก์เลส จากที่ได้กล่าวมาเกี่ยวกับโครงสร้างของ คลัสเตอร์ ส่วนหนึ่งที่สำคัญของระบบ คือ ประสิทธิภาพการติดต่อผ่านเครือข่าย มีผลต่อการทำงานของระบบคลัสเตอร์เป็นอย่างมาก โดยเฉพาะอย่างยิ่งระบบคลัสเตอร์ที่มีขนาดใหญ่ จะต้องมีการออกแบบเครือข่ายให้สามารถรองรับการขนถ่ายข้อมูลได้ทีละมาก ๆ ดังตัวอย่างในบทที่ 2

จากเหตุดังกล่าวจึงไม่นิยมติดตั้งระบบ ดิสก์เลสในระบบขนาดใหญ่ที่ต้องการการใช้งานเครือข่ายที่มีประสิทธิภาพสูง แต่ในระบบที่เราทำการศึกษา เรามุ่งเน้น การทดสอบระบบที่มีขนาดเล็ก ขนาด 4-8 เครื่อง ซึ่งขนาดของขีดความสามารถในการขนถ่ายข้อมูลในเครือข่ายที่ใช้มีอย่างเหลือเฟือ ทำให้สามารถนำมาใช้ได้ และการนำมาใช้ยังทำให้ ลดค่าใช้จ่ายดังที่ได้กล่าวมาแล้วข้างต้น

4.3 การทำงานของ NIS

NIS (Network Information System) เป็นชื่อของระบบเครือข่ายที่ถูกพัฒนาขึ้นโดยบริษัท Sun Microsystem รูปแบบของ NIS เป็นระบบที่ออกแบบให้ เครื่องลูกทุกเครื่องในเครือข่ายสามารถทราบถึงค่าปรับแต่ง (config) ของทุกเครื่องทั้งระบบได้ ผู้ใช้ที่ขอเข้าใช้งานจากเครื่องใดก็ตามที่อยู่ในระบบสามารถเข้าใช้งาน ได้โดยผ่านรหัสผ่านเดียวที่ถูกจัดเก็บไว้ที่เครื่องกลาง ไม่เพียงเฉพาะ รหัสผ่าน ระบบ NIS ยังสามารถเก็บข้อมูลเฉพาะต่างๆของแต่ละผู้ใช้ เมื่อผู้ใช้ทำการ ขอเข้าใช้งานจากเครื่องใดๆ ก็จะได้ค่าติดตั้งเหล่านั้นติดไปด้วย โครงสร้างของ NIS นั้นมีลักษณะคล้ายกับระบบ โดเมนเนม ที่ใช้งานอยู่ในปัจจุบัน แต่เป็นระบบที่ใช้งานอยู่ในเฉพาะ เครือข่ายเล็กๆเท่านั้น

NIS ใช้รูปแบบการทำงานแบบ ไคลเอ็นต์-เซิร์ฟเวอร์ และใช้ RPC (Remote Procedure Call) ในการสื่อสารกันระหว่างเครื่องในระบบ

ฐานข้อมูลของ NIS นั้นจะเก็บในรูปแบบที่เรียกว่า DBM format ซึ่งมีความซับซ้อนมากกว่าการเก็บข้อมูลเป็น ASCII มาก ตัวอย่างเช่น ไฟล์ /etc/passwd และไฟล์ /etc/group ในระบบยูนิกส์สามารถแปลงให้อยู่ในรูปแบบของ DBM ได้โดยใช้ โปรแกรมที่ช่วยในการแปลง ASCII ให้อยู่ในรูป DBM เช่น โปรแกรม makedbm ที่มากับลีนุกส์ สามารถทำงานดังกล่าวได้ เครื่องกลางของระบบ NIS จะต้องมีข้อมูลดังกล่าว ทั้งในรูปของ ASCII และที่อยู่ในรูป DBM

4.4 การทำงานของ NFS

NFS (Network File System) เป็นระบบไคลเอ็นต์-เซิร์ฟเวอร์ ที่สามารถทำให้เครื่องใด ๆ ที่ต้องการสามารถเข้าอ่าน หรือเขียนข้อมูลบนเครื่องที่อยู่ปลายทางได้เสมือนเป็นเขียนบนเครื่องของตน โดยเครื่องลูกที่ใช้ จะต้องทำการติดตั้ง ไคลเอ็นต์ของ NFS ส่วนเครื่องอื่น ๆ ที่เราต้องการขอเข้าใช้งานเราจะต้องลง NFS เซิร์ฟเวอร์ โดยเครื่องทั้งสอง ต้องสนับสนุนการใช้งานโพรโตคอล TCP/IP

NFS ถูกพัฒนาขึ้นโดยบริษัท Sun Microsystem ซึ่งได้ออกแบบมาตรฐานของระบบไฟล์เซิร์ฟเวอร์ ในยุคแรก ๆ การทำงานของ NFS จะทำงานโดยใช้ RPC (Remote Procedure Call) เป็นตัวช่วยในการติดต่อขนย้ายข้อมูลระหว่างเครื่องคอมพิวเตอร์

การใช้งาน NFS ผู้ใช้ หรือผู้ดูแลระบบ สามารถใช้คำสั่ง mount ทำการเรียกใช้งาน ไปยัง พาธที่ถูกต้องบนเครื่องที่ต้องการได้ โดย การใช้งาน เมื่อทำการ เมาท์พาธที่ต้องการจากเครื่องที่ได้รับอนุญาตมายังเครื่อง ที่ต้องการแล้ว จะสามารถใช้ ไดเรคทรอรี่ที่เมาท์ได้โดยเสมือนว่า ไดเรคทรอรี่ที่เมาท์มานั้น อยู่ที่เครื่องที่เราใช้งานอยู่ โดยจะสามารถอ่านได้อย่างเดียวหรือว่า ทั้งเขียนและอ่านได้ ก็ขึ้นกับการกำหนดจากฝั่งเซิร์ฟเวอร์ว่าจะให้เครื่องนี้ เข้าถึงข้อมูลได้เพียงใด

RPC (Remote Procedure Call)

คือโพรโตคอล ที่อนุญาตให้โปรแกรมสามารถ เรียกขอข้อมูลไฟล์ หรือโปรแกรมผ่านระบบเครือข่ายโดยไม่ต้อง เข้าใจถึงรายละเอียดของระบบเครือข่าย โดยใช้เทคโนโลยีของ lightweight processes หรือที่เรียกว่า thread (เธรด)

4.5 โครงสร้างการทำงานของ PVM (Parallel Virtual Machine)

4.5.1 ภาพรวมของ PVM

PVM เป็นซอฟท์แวร์ที่ทำหน้าที่จัดแบ่งการทำงานออกเป็น เฟรมเวิร์ก (framework) ซึ่งจะต้องเป็นโปรแกรมที่สามารถทำงานเป็น แบบขนาน และจะต้องเหมาะสมกับฮาร์ดแวร์ ที่มีอยู่ PVM จะสามารถมองกลุ่มของคอมพิวเตอร์ที่แตกต่างกันซึ่งเชื่อมต่อกันเป็นระบบเสมือนเป็นเครื่องเดียว นอกจากนี้ PVM ยังสามารถควบคุมการหาเส้นทางการส่งข้อมูล การแปลงข้อมูล และจัดตารางงานบนเครือข่าย ที่ประกอบขึ้น ด้วยคอมพิวเตอร์ที่มีสถาปัตยกรรมที่แตกต่างกัน

PVM computing model เป็นโครงสร้างการเขียนโปรแกรมที่สามารถปรับให้เข้ากับโปรแกรมโดยที่ผู้ใช้ จะต้องเขียนโปรแกรมจัดโครงสร้างของคำสั่ง ให้แบ่งออกเป็นงานออกเป็นส่วน ๆ ที่ทำงานร่วมกัน โดยที่แต่ละงานจะต้องทำงานร่วมกับ PVM library routine จะต้องใช้เพื่อทำการเริ่มต้นและหยุดการติดต่อของการส่งถ่ายงานบนเครือข่าย ในการติดต่อสื่อสารอาจจะต้องมีการส่งแบบ broadcast , barrier synchronization , global sum

4.5.2 ระบบของ PVM

PVM (Parallel Virtual Machine) เป็นผลิตภัณฑ์อย่างหนึ่งของการทำการวิจัยการประมวลผลผ่านระบบเครือข่าย โดยวัตถุประสงค์ของการทำการวิจัยนี้คือ ทำการสืบค้นหาประเด็นของปัญหาที่เกิดขึ้นและคิดหาวิธีแก้ไขปัญหาต่าง ๆ โดย PVM นี้จะรวมถึงเซตของ Software tool และ ไลบรารี ต่าง ๆ เข้าไปเพื่อให้มีความสะดวกและความยืดหยุ่นในการใช้งานทั่วไป และโดยวัตถุประสงค์โดยรวมของ PVM System คือ สามารถทำให้กลุ่มของเครื่องคอมพิวเตอร์ที่มีสถาปัตยกรรมที่ต่างกันสามารถทำงานร่วมกันได้ในลักษณะของกระประมวลผลแบบขนาน

คุณสมบัติที่สำคัญที่เป็นพื้นฐานระบบของ PVM จะมีดังนี้

- User-configured host pool : โปรแกรมที่จะถูกรัน นั้นจะถูกกำหนดโดยผู้ใช้ว่าจะป้อนให้กับกลุ่มของเครื่อง (Machine) กลุ่มใดก็ได้ โดยกลุ่มเครื่องนั้นๆ ต้องทำการรัน อยู่ภายใต้สภาวะที่เหมาะกับการใช้งาน PVM ด้วย ซึ่งกลุ่มเครื่อง นั้นอาจประกอบด้วย ซีพียูเดียว (Single-CPU) หรือ หลายซีพียู (Multiprocessor) (รวมถึง Shared-memory และ Distributed-memory) ที่อาจเป็นส่วนหนึ่งของกลุ่มของเครื่องนั้นก็ได้ และกลุ่มเครื่องที่เราเลือกนั้นเราสามารถทำการเพิ่มหรือลดจำนวนเครื่องได้ด้วย ในระหว่างการทำงาน (จุดนี้เองที่เป็นคุณสมบัติที่สำคัญในการทำ Fault tolerance)

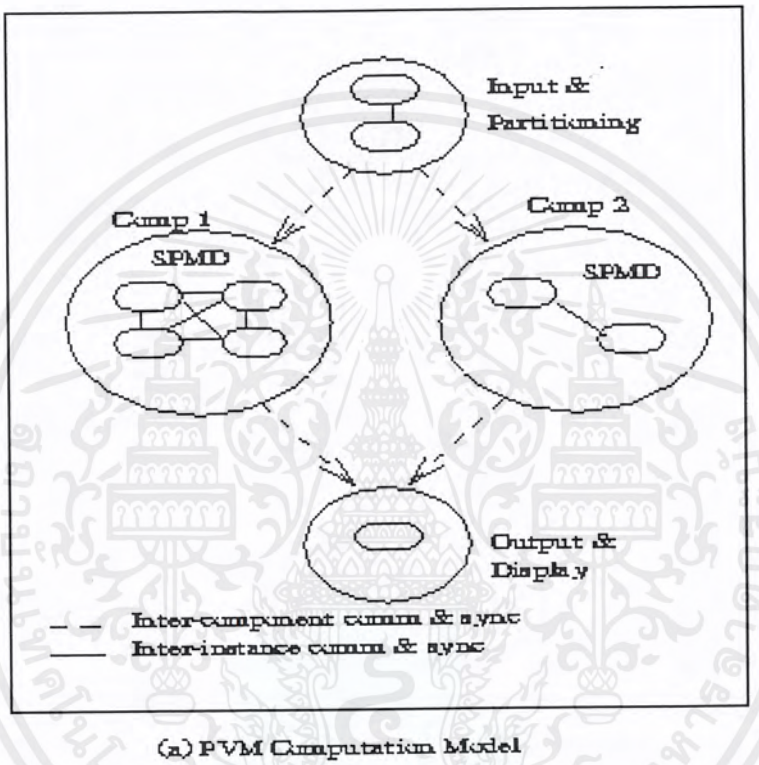
- Translucent access to hardware : โปรแกรม อาจมอง Hardware environment เป็น Attributeless collection ของ Virtual processing elements หรืออาจเลือกคุณสมบัติของคอมพิวเตอร์ ในกลุ่มเครื่องที่ทำงานที่เหมาะกับ โปรแกรมนั้นที่สุด
- Process-based computation : หน่วยการทำงานแบบขนานใน PVM คือ Task (ในบางส่วนแต่ไม่จำเป็นเสมอไปสำหรับ Unix process) ในการควบคุมการทำงานนั้นต้องมีการควบคุมในส่วนของการติดต่อสื่อสารและการประมวลผล และจะไม่มีการบังคับลงไปว่าให้ โพรเซสใดทำงานอยู่บน โพรเซสเซอร์ ใดจัดการ ซึ่งจริง ๆ แล้วบางทีทาส์ก ต่างๆอาจถูกกระทำด้วยโพรเซสเซอร์ เพียง 1 ตัวเท่านั้นก็ได้
- Explicit message-passing model : ข้อมูลต่างๆที่จะทำการคำนวณจะถูกรวบรวมไว้ ซึ่งในข้อมูลเหล่านั้นจะมีส่วนที่เป็น ข้อมูล, ฟังก์ชัน หรือ Hybrid decomposition ในการทำงานร่วมกันของกลุ่มคอมพิวเตอร์ นี้จะใช้การส่งและการรับข้อมูลในรูปแบบของเมสเซจ จากเครื่องหนึ่งไปยังอีกเครื่องหนึ่ง โดยขนาดของเมสเซจ นี้จะขึ้นอยู่กับขนาดหน่วยความจำที่ยังเหลืออยู่
- Heterogeneity support : ระบบของ PVM นั้นสนับสนุนเครื่องโหนด ที่มีความแตกต่างกัน ไม่ว่าจะเป็นด้านของ เน็ตเวิร์ก , โปรแกรมที่ทำงาน โดยต้องสามารถทำ ส่งเมสเซจ ได้โดย PVM นั้นยอมให้เมสเซจ สามารถเก็บชนิดของข้อมูล ได้มากกว่า 1 ชนิด และชนิดของข้อมูลจะถูกเปลี่ยนให้เหมาะสมกับเครื่องที่จะรับเมสเซจ นั้นด้วย
- Multiprocessor support : PVM นั้นใช้วิธีการส่งเมสเซจ ดังนั้นจึงเป็นการทำงานกับหลายๆ โพรเซสเซอร์ อยู่แล้วซึ่งเป็นข้อดีที่สนับสนุนทางด้านฮาร์ดแวร์ ซึ่งมีผู้ขายหลายรายที่พยายามทำผลิตภัณฑ์ ของตนให้มีคุณสมบัติตามนี้

ในระบบของ PVM นั้นจะประกอบด้วย 2 ส่วน

ส่วนแรก เป็น เดมอน (Daemon) ชื่อ pvmd3 หรืออาจย่อเป็น pvmd (ตัวอย่าง เดมอนของโปรแกรมเมลล์ ก็เป็นเดมอน ซึ่งทำงานอยู่เบื้องหลัง คอยทำหน้าที่รับและส่งจดหมาย) ซึ่งจะต้องมีอยู่บนคอมพิวเตอร์ เครื่องที่เราทำการจัดขึ้นมาให้เป็นกลุ่มของ Virtual machine โดย pvmd3 สามารถทำการติดตั้ง ลงไปได้โดยผู้ใช้ที่มีสิทธิในการใช้ (Valid login) และมีความต้องการใช้ งาน PVM ซึ่งเป็นจุดเริ่มต้นในการสร้าง Virtual machine ขึ้นมา โดย การเรียกใช้งาน PVM นั้นจะสามารถเรียกใช้ได้จาก Unix prompt ของทุก ๆ เครื่อง

ส่วนที่สอง เป็นส่วนของไลบรารี ที่จัดการการติดต่อกับระบบ PVM ซึ่งจะบรรจุด้วยการขั้นตอนทำงานต่างๆในตอนเริ่มต้นที่จำเป็นสำหรับการทำงานร่วมกันของทาส์ก, โปรแกรม และบางส่วนของไลบรารี ยังมีส่วนที่ผู้ใช้สามารถเรียกใช้ได้สำหรับการทำ Message passing , การแตกโพรเซส , การรวมทาส์ก ต่างๆ , และการปรับปรุง Virtual machine

โครงสร้างของการทำงานแบบ PVM มีพื้นฐานมาจากความคิดที่ว่าการทำงานของโปรแกรมประกอบด้วยทาส์กจำนวนมาก ในแต่ละทาส์กที่เก็บโปรแกรม ที่จะต้องทำการคำนวณนั้นก็จะเป็น การทำงานที่ใช้ทรัพยากรของระบบ ในบางครั้งโปรแกรม จะถูกจัดการแบบขนาน ทั้งฟังก์ชันการทำงาน (ข้อมูลแต่ละส่วนอาจถูกจัดการในฟังก์ชัน การทำงานที่ต่างกัน) ซึ่งรูปแบบการทำงานแบบนี้คล้ายกับการทำงานแบบ SPMD(single-program multiple-data) model และก็อาจมีการทำงานแบบ SPMD นี้ซ้อนกันอยู่อีกก็ได้ดังแสดงในรูปที่4.1 ซึ่งจะเห็นได้ว่า PVM สามารถรองรับการทำงานนี้ได้ และสถาปัตยกรรมของ PVM ที่สามารถรองรับอุปกรณ์ที่มีความแตกต่างกันมาต่อเป็นระบบขึ้นมาดังในรูปที่ 4.1

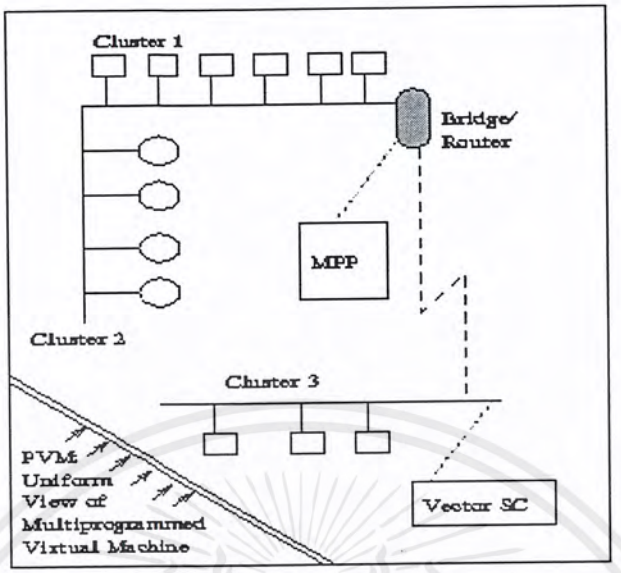


รูปที่ 4.1 ภาพรวมของระบบ PVM (PVM System Overview)

ระบบ PVM นั้นสนับสนุนภาษา C ,C++ และ Fortran และกลุ่มของ Language interface ก็รวมอยู่ในภาษากลุ่มนี้ด้วย จึงเป็นที่น่าสังเกตว่าโปรแกรม ส่วนใหญ่จะถูกเขียนขึ้นมาด้วยภาษา C และ Fortran ดังนั้นจึงมีความจำเป็นที่จะต้องมีพื้นฐานเกี่ยวกับการเขียนโปรแกรมเชิงวัตถุ และการใช้ เมธอด

ภาษา C และ C++ จะถูกรวมเข้ากับไลบรารีของการใช้งาน PVM และถูกแสดงออกมาในรูปแบบของฟังก์ชัน ซึ่งเป็นรูปแบบของระบบพวกระบบที่ใช้ C เป็นพื้นฐาน รวมถึงยูนิกส์ ด้วย และ ฟังก์ชันการทำงานก็จะมีการส่งค่าเข้าฟังก์ชัน ที่ต้องรวมเข้าไปด้วย ซึ่งอาจเป็นค่าของ พารามิเตอร์ หรือ พอยนต์เตอร์ ก็ได้และผลที่ได้จากฟังก์ชัน คือค่าที่จะส่งคืนกลับออกมา และยังมีการใช้มาโคร ซึ่งใช้เป็นค่าคงที่ของระบบและบางทีก็อาจเป็นตัวแปรของระบบ เช่น errno และ pvm_errno Application หรือ

โปรแกรมใด ๆ ที่เขียนขึ้นด้วย C ,C++ สามารถใช้ PVM ไลบรารี ได้ด้วยโดยทำการ เรียกไลบรารี ไว้ในตอนเริ่มโปรแกรมเท่านั้น



(b) PVM Architectural Overview

รูปที่ 4.2 การเชื่อมต่อของสถาปัตยกรรมต่างๆในระบบ PVM

ส่วนถ้าเป็นภาษา Fortran ก็จะมี ไลบรารี ให้เรียกใช้ได้เหมือนกันแต่การเรียกใช้จะต้องเรียกใช้แบบ Subroutine แทนการเรียกแบบฟังก์ชัน ที่เป็นแบบนี้เพราะคอมไพเลอร์ บางตัวนั้นไม่เข้าใจอินเตอร์เฟส ของภาษา Fortran ที่เป็นรูปแบบฟังก์ชัน และในไลบรารี เราสามารถรู้ได้ว่าไลบรารี นี้เป็นของภาษาใดโดยสังเกต Prefix ไลบรารี ของภาษา C จะขึ้นด้วย pvm และถ้าเป็น Fortran จะขึ้นด้วย pvmf

งานทุกอย่างที่ทำบน pvm จะถูกกำหนดเป็นตัวเลข Integer และเราเรียกตัวเลขนั้นว่าเป็น Task Identifier (TID) โดย เมสเซจจะที่ต้องการส่งหรือรับนั้นต้องอาศัย TID นี้ และ TID ของ Virtual Machine จะต้องไม่ซ้ำกัน ซึ่งตัวเลขเหล่านี้จะถูกกำหนดโดย Local pvmd โดยที่ผู้ใช้ ไม่สามารถเลือกได้

และเรายังมีการทำเป็นกลุ่มของทาส์กได้ด้วย โดยผู้ใช้สามารถระบุเบอร์ของทาส์กในกลุ่มของตนได้และเมื่อมีทาส์ก ผ่านเข้ามาทาส์ก นั้นจะถูกแทนค่าเป็นตัวเลขค่าหนึ่งซึ่งไม่ซ้ำกับเบอร์ของทาส์กในกลุ่มนั้นซึ่งทาส์ก ที่เข้ามาใหม่นี้จะไม่มีผลกระทบต่อการทำงานเดิมทำให้เราสามารถกำหนด กลุ่มของทาส์กต่างๆเกิดการซ้อนกันได้ทำให้เราสามารถเลือกกลุ่มในการทำงานได้อย่างเป็นอิสระซึ่งการทำงานแบบนี้จะอยู่ในส่วนการใช้งานของกลุ่มของฟังก์ชัน ซึ่งสามารถเรียกใช้ได้จาก libpvm3.a และในรายละเอียดของการทำงานเราจะไม่กล่าวถึง

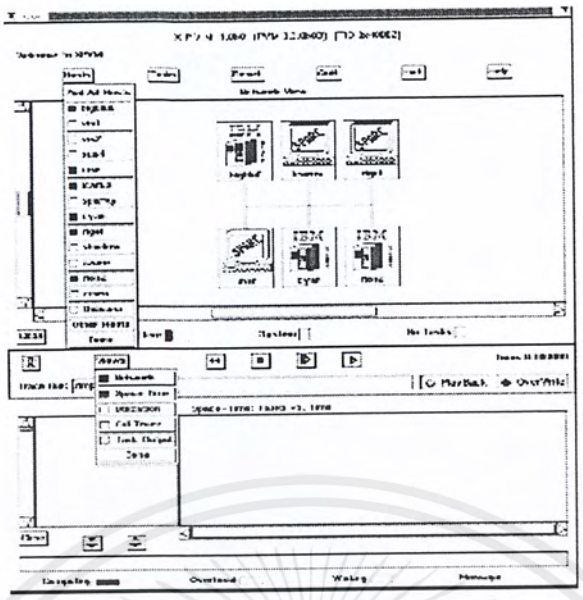
ในการเขียนโปรแกรมที่เราต้องการใช้ฟังก์ชัน การทำงานของ PVM นั้นเราสามารถทำได้โดยเขียนโปรแกรมด้วยภาษา C , C++, Fortran77 และทำการฝังไลบรารีของ PVM ลงไปในโค้ด ซึ่งเมื่อเราทำการคอมไพล์จนเป็นโปรแกรม ขึ้นมาแล้วทำการรันบนกลุ่มของโฮสต์ ที่มีการทำงานแบบ PVM Program นั้นจะถูกแบ่งทาส์กออกเป็นส่วนต่าง ๆ แล้วทำการกระจายไปบนกลุ่มโฮสต์นั้น (Virtual Machine) ซึ่งจะถูกส่งออกไปในรูปของเมสเซจและภายใน Virtual machine จะทำการแลกเปลี่ยนเมสเซจ เพื่อทำการ เรียก

ใช้งาน โปรแกรมนั้นโดยใช้หลักการของ Message passing และในระหว่างการเรียกใช้โปรแกรมสามารถมีทาส์กอื่นเข้ามาได้ด้วย ซึ่งในการจัดการงานต่าง ๆ นั้นจะมีค่า TID ในการระบุว่างานนี้มาจากที่ใดและจะถูกส่งไปที่ใด

สถาปัตยกรรมต่างๆที่ pvm3 สนับสนุน

PVM_ARCH	Machine	Notes
AFX8	Alliant FX/8	
ALPHA	DEC Alpha	DEC OSF-1
BAL	Sequent Blance	DYNIX
BFLY	BBN Butterfly TC2000	
BSD386	80386/486 PC running Unix	BSDI, 386BSD, NetBSD
CM2	Thinking Machines CM2	Sun front-end
CM5	Thinking Machines CM5	Uses native messages
CNVX	Convex C-series	IEEE f.p.
CNVXN	Convex C-series	Native f.p.
CRAY	C-90, YMP, T3D port available	UNICOS
CRAY2	Cray-2	
DGAV	Data General Aviion	
E88K	Encore 88000	
HP300	HP-9000 model 300	HPUX
HPPA	HP-9000 PA-RISC	
I860	Intel iPSC/860	Uses native messages
IPSC2	Intel IpSC/2 386 host	Sys V, Uses native messages
KSR1	Kendall Square KSR-1	OSF-1, uses shared memory
LINUX	80386/486 PC running Unix	LINUX
MASPAR	Maspar	DEC front-end
MIPS	MIPS 4680	
NEXT	NeXT	
PGON	Intel Paragon	Uses native messages
PMAX	DEC station 3100, 5100	Ultrix
RS6K	IBM RS6000	AIX3.2
RT	IBM RT	
SGI	Silicon Graphics IRIS	IRIX 4.x
SGI5	Silicon Graphics IRIS	IRIX 5.x
SGIMP	SGI multiprocessor	Uses shared memory
SUN3	Sun 3	SunOS 4.2
SUN4	Sun 4, SPARC station	SunOS 4.2
SUN4SOL2	Sun 4, SPARC station	Solaris 2.x
SUNMP	SPARC multiprocessor	Solaris 2.x, uses shared memory
SYMP	Sequent Symmetry	
TITN	Stardent Titan	
U370	IBM 370	AIX
UVAX	DEC Micro VAX	

ตารางที่ 4.1 แสดงสถาปัตยกรรมที่สนับสนุน PVM (PVM_ARCH names used in PVM 3)



รูปที่ 4.3 หน้าจอแสดงผลแบบกราฟิกของ XPVM (XPVM system adding hosts)

4.5.3 ขั้นตอนการติดตั้ง PVM

ก่อนที่จะทำการติดตั้ง PVM จะต้องทำการตั้งต่าตัวแปรระบบตามที่กำหนด ได้แก่ PVM_ROOT ให้เป็นสถาปัตยกรรมของเครื่อง เช่น. "CRAY", "LINUX" และตั้งค่า PVM_ROOT ให้ชี้ไปยังที่อยู่ของ PVM เช่น SHOME/pvm3 หรือถ้าใส่ไว้ในระบบเช่น /usr/local/pvm3 ตัวอย่างเป็นไฟล์ .bashrc ใน bash ทำดังนี้

```
PVM_ROOT=SHOME/pvm3
export PVM_ROOT
```

ทำการติดตั้ง PVM เข้ายังระบบ โดยพิมพ์คำสั่ง "make" เพื่อสร้าง pvmd3 และ C ไลบรารี (libpvm3.a) และ Fortran library(libfpvm3.a) และโปรแกรมที่รันบนคอนโซล(pvm)

4.5.4 การเริ่มใช้งานและหยุดการใช้งาน PVM

เริ่มต้นการใช้งาน PVM โดยการเรียกโปรแกรม SPVM_ROOT/lib/pvm โปรแกรมนี้จะเรียกคอนโซลขึ้นมาเพื่อเรียก pvmd และทำการ เพิ่มเครื่องลูกข่ายที่ต้องการเข้าร่วม โดยใช้ คำสั่ง "add" ได้โดยตรง หรือสามารถใส่ชื่อเครื่องไปในไฟล์ที่ต้องการแล้วเรียก การทำงานพร้อมๆกันได้โดยใช้ คำสั่ง pvmd ตามด้วยไฟล์ที่ใส่ชื่อเครื่อง ดังเช่น

$PVM_ROOT/lib/pvmd my_hosts

การหยุดการใช้งาน PVM ได้โดยพิมพ์คำสั่ง "halt" ที่คอนโซลของ PVM หรือในโปรแกรมสามารถเรียกฟังก์ชัน pvm_halt() ได้หรือสามารถสั่ง kill โพรเซสของ pvmd3 ได้ทันทีที่ต้องการปิด แต่การปิดโดยวิธี kill อาจทำให้เกิดปัญหาได้

4.5.5 คำสั่งที่ใช้งานบนคอนโซลของ PVM

เป็นคำสั่งในการควบคุมการทำงานของ PVM ซึ่งสามารถป้อนได้ในขณะที่เป็น pvm console ซึ่งจะเป็นลักษณะดังนี้

pvm >

ซึ่งมีคำสั่งดังนี้

add	ทำการ add host ให้เป็น Virtual machine
alias	define or lists command aliases
conf	lists the configuration of the virtual machine including hostname , pvmd task ID , Architecture type and relative speed rating
delete	ทำการ delete host name ออกจาก Virtual machine และ PVM process ที่กำลังทำงานอยู่ที่ host นั้นจะหายไปด้วย
echo	echo arguments
halt	kills all PVM processes including console ,and then shuts down PVM. All demons exit
help	get information about any of interactive command
id	prints the console task id.
jobs	lists running jobs
kill	terminate any PVM process
mstat	shows the status of specified hosts
ps –a	lists all processes in virtual machine
pstat	shows the status of a single PVM process
quit	exits the console but PVM jobs still running
reset	kills all PVM processes except consoles ,and resets all the internal PVM tables and message queues. The demons are left in an idle state
setenv	display or sets environment variables

sig	sends the signal to the task
spawn	starts a PVM application ซึ่งมี Option ในการใช้มากมาย ซึ่งจะไม่กล่าวในที่นี้
trace	sets or displays the trace event mask
unalias	undefined command alias
version	prints version of PVM being used

4.5.6 Host file Options

เป็น Option ที่สามารถใช้ใน file Hostfile ซึ่งช่วยในการ Setup กลุ่ม host ที่เราต้องการทำเป็น Virtual machine ซึ่งจะมี option ต่างๆดังนี้

lo = userid allows you to spacify an alternative login name for this host

so = pw will cause PVM to prompt for password on this host. This is useful in the cases where you have a different userid and password on a remote host

dx = location of pvmd
allows you to specify a location other than the default for this host. This is useful if you want to use your own personal copy of pvmd

ep = paths to user executables
allows you to specify a series of path for application tasks

sp = value specifies the relative computational speed of the host compared with other hosts in the configuration. The range of possible values is 1 to 1000000 with 1000 as the default

bx = location of debugger
specifies which debugger script (if debugger is requested in the spawn routine) โดยจะมี environment ชื่อ PVM_DEBUGGER เซตค่าไว้เป็น pvm3/lib/debugger

wd = working directory
specifies a working directory in which all spawned tasks on this host will execute ซึ่งค่า default คือ SHOME

ip = hostname specifies an alternate name to resolve to the host IP address

so = ms specifies that a slave pvmd will be started manually on this host

ซึ่งตัวอย่างของ hostfile จะเป็นดังนี้

```
# Comment lines start with # (blank lines ignored)
gstws
ipsc      dx=/usr/geist/pvm3/lib/I860/pvmd3
ibm1.scri.fsu.edu lo=gst so=pw

# set default options for following hosts with *
* ep=$sun/problem1:~/nla/mathlib
sparky
# azure.epm.ornl.gov
midnight.epm.ornl.gov

# replace default options with new values
*lo=gageist so=pw ep=problem1
thud.cs.utk.edu
speedy.cs.utk.edu

# machines for adding later are specified with &
# these only need listing if option are required
&sun4   ep=problem1
&castor dx=/usr/local/bin/pvmd3
&dasher.cs.utk.edu        lo=gageist
&elvis   dx=~/pvm3/lib/SUN4/pvmd3
```

รูปที่ 4.4 แสดงการเขียน hostfile ที่ใช้คำสั่งต่างๆ

4.5.7 จุดประสงค์หลักของ PVM3

- Fault tolerance ซึ่งจะไม่เป็นแบบ Automatically recover แต่จะใช้วิธี polling เพื่อที่จะบอกว่าสภาพของระบบเป็นอย่างไรซึ่งเมื่อโฮสต์ ภายใน virtual machine นั้นเกิดทำงานไม่ได้ขึ้นมา PVM จะรายงานมาและให้เราจัดการปรับแต่งค่าติดตั้งบนเครื่อง โดยการ delete host และ add host ใหม่เข้าไปแทน
- Scalability ใน virtual machine ที่เราสามารถสร้างขึ้นนั้นสามารถมี โฮสต์ ได้เป็นร้อย ๆ และสามารถทำงานกว่าพัน ๆ งานได้
- Heterogeneity คือ มีความสามารถในการทำงานรวมกันของอุปกรณ์ชนิดต่าง ๆ ได้
- Portability มีความสามารถในการย้ายโปรแกรมจากเครื่องหนึ่งไปทำงานกับเครื่องอื่น ๆ ได้ ซึ่งมีการทำงานที่คล้ายกับระบบยูนิกส์

ในการติดต่อกันของโฮสต์ ต่างๆใน Virtual machine นั้นจะใช้ Socket ในการติดต่อและใช้ Protocol TCP และ UDP และต้องการ IP ในการติดต่อด้วย และถ้าบางโฮสต์ ไม่ได้ใช้ Socket ในการติดต่อสื่อสารแต่ใช้ Front-end ก็สามารถเชื่อมต่อเป็น Virtual machine ได้ด้วย

4.6 โครงสร้างการทำงานของ MPI (Message Passing Interface)

4.6.1 ความรู้เบื้องต้นเกี่ยวกับ MPI

MPI (Message-Passing Interface) คือรูปแบบของมาตรฐานของการทำงานแบบ message-passing โดย Message passing คือรูปแบบที่งานอย่างกว้างขวางในการทำ parallel machines ซึ่งอำนวยความสะดวกในการพัฒนาของโปรแกรมประยุกต์ที่เป็นแบบขนาน MPI สามารถทำงานได้ทั้งภาษา C และ FORTRAN 77 จุดมุ่งหมายของการออกแบบ MPI ก็เพื่อความสะดวก มีประสิทธิภาพและทำงานได้หลายหน้าที่ โดย MPI สามารถทำงานได้กับเครื่องตั้งแต่ระบบ MPP (Massive-Parallel Machine) ไปจนถึงระบบเน็ตเวิร์คของเครื่องเวิร์กสเตชั่น

MPI มีคุณสมบัติมากมาย เช่น ทำงานแบบ Point-to-Point , การสื่อสารสนับสนุนในแบบซิงโครนัส หรือ อะซิงโครนัสและสนับสนุน หน่วยประมวลผลเฉพาะการสื่อสาร (Communication Processor) เป็นต้น

ข้อดีของมาตรฐานการทำงานแบบ message-passing คือมีการใช้งานทรัพยากรในการทำงานค่อนข้างน้อย และง่ายต่อการนำไปใช้งาน ในการทำงานแบบ distributed-memory ทำให้มีประสิทธิภาพ และยิ่งไปกว่านั้น มาตรฐานของ message-passing ที่ได้กล่าวมาข้างต้นยังมีผลทำให้มีการพัฒนาอย่างมีประสิทธิภาพรวมถึง สร้างอุปกรณ์ต่างๆเพื่อสนับสนุนการทำงานให้มีเสถียรภาพ เมื่อได้แนวคิดที่ถูกต้องชัดเจน

มาตรฐานการทำงานของ MPI มีจุดมุ่งหมายเพื่อการพัฒนาการเขียนโปรแกรมและการใช้งานระบบแบบขนาน โดยมีการออกแบบให้สามารถใช้งานเบื้องต้นกับภาษา Fortran 77 และ C เป็นพื้นฐาน โดยผู้โปรแกรมต้องมีความเข้าใจและสามารถออกแบบ โปรแกรมให้ทำงานแบบขนานได้ เพื่อที่จะทำให้การทำงานมีประสิทธิภาพสูง

การออกแบบโปรแกรมให้ใช้งาน MPI ได้อย่างมีประสิทธิภาพนั้น จะต้องศึกษาถึงฟังก์ชันต่างๆอย่างลึกซึ้ง โดยมาตรฐานของ MPI นั้นกำหนดให้สามารถผู้ใช้หรือโปรแกรมที่ผู้ใช้เขียนขึ้นนั้น ควบคุมการส่งเมสเซจไปควบคุมต่าง ๆ เช่นการใช้งานหน่วยความจำในเครื่องโหนดในระบบคลัสเตอร์ โดยตั้งแต่มิถุนายนปี 1994 MPI ได้เข้ามามีบทบาทต่อการทำงานของระบบคลัสเตอร์มีการนำมาใช้งานอย่างแพร่หลาย มีการสร้างโปรแกรมมาใช้งานในด้านต่าง ๆ

อีกจุดมุ่งหมายซึ่งสำคัญคือ การพยายามทำให้สามารถใช้งาน MPI ในระบบที่มีเครื่องหลายสถาปัตยกรรมได้โดยปัจจุบันยังไม่สามารถทำได้อย่างเต็มรูปแบบ โดยสามารถใช้งานได้เฉพาะ fortran แต่นั่นก็แสดงว่า การทำงานของ MPI สามารถแปลงเมสเซจให้เครื่องต่าง สถาปัตยกรรม สามารถเข้าใจถึงภาษาเดียวกันได้ อย่างไรก็ตาม การปรับแต่งระบบให้มีความเหมาะสมต่องานที่ระบบนั้น ๆ ใช้งานอยู่ก็มีความสำคัญไม่ยิ่งหย่อนต่อกัน เพื่อที่จะให้ได้มาซึ่งประสิทธิภาพที่น่าพอใจ โดยทั่วไปแล้วการทำงานของ MPI จะเป็นการควบคุมการใช้งานหน่วยความจำแบบกระจาย (distributed-memory) บนระบบคลัสเตอร์ซึ่งสามารถทำงานร่วมกันบนระบบคลัสเตอร์ได้ บนระบบทั่วไปที่ต่อเครื่องโหนดต่าง ๆ ผ่านระบบผ่านเครือข่ายใด ๆ ก็ตาม และแม้การทำงานบนเครื่อง ๆ เดียวก็สามารถทำงาน ได้อย่างมีประสิทธิภาพ โดยระบบที่ทำงานมีเสถียรภาพสูง

ความสามารถของ MPI สามารถทำงานได้บนระบบคลัสเตอร์ขนาดใหญ่ที่มีความ หลากหลายของสถาปัตยกรรม โดยเสมือนทำงานอยู่บนระบบขนาดเล็กทั่วไป โดย MPI สามารถส่งเมสเซจไปยังเครื่องโหนดได้ โดยที่ผู้ใช้ไม่ต้องคำนึงถึง การเปลี่ยนแปลงข้อมูลให้เข้าเข้ากับการทำงานของสถาปัตยกรรมนั้น ๆ โดยส่วนนี้ การทำงานของ MPI จะช่วยจัดการส่วนนี้ให้อย่างมีประสิทธิภาพ แต่อย่างไรก็ตาม MPI ที่ต้องการใช้งานบนระบบที่มีความหลายหลายของสถาปัตยกรรม จะต้องใช้เวอร์ชันของ MPI ที่มีปรับแต่งให้มีความสามารถทำงานดังกล่าวได้ ไม่สามารถใช้ เวอร์ชันทั่วไปที่ใช้งานอยู่

ความสำคัญในการออกแบบจุดมุ่งหมายของ MPI ให้สามารถใช้งานบนหลายๆเครื่องที่มีความแต่ต่างด้านองค์ประกอบได้นั้น เช่น ความเร็วของซีพียู ขนาดของหน่วยความจำ หรือ แม้กระทั่งความเร็วของเน็ตเวิร์คอินเตอร์เฟส จากปัญหาดังกล่าวโครงสร้างของ MPI จะไม่มองถึงส่วนการทำงานของแต่ละเครื่องโหนดว่าจะทำงานอย่างไร จัดการอย่างไร แต่ MPI พยายามที่เน้นความสนใจไปที่ผลลัพธ์ที่ได้จาก การคำนวณ ของเครื่องโหนดแต่ละเครื่อง MPI มีความสามารถทำงานในการส่งเมสเซจ รับเมสเซจ โดยไม่ผ่านบัฟเฟอร์

โดยทั่วไปแล้ว MPI เป็นมาตรฐานที่มีความเฉพาะตัว เหมาะกับการนำมาใช้ ในทำงานในระบบคลัสเตอร์ มีไลบรารี ที่มีความสามารถเฉพาะด้านในหลายๆด้าน สามารถนำมาใช้เป็นตัวช่วยในการเขียนโปรแกรมแบบขนานได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น มีการตรวจสอบการสูญหายหรือความสมบูรณ์ของเมสเซจที่ได้ส่งไปยังเครื่องโหลด หาเกิดปัญหา จะมีการแก้ปัญหาได้ด้วยความสามารถของ MPI ทำให้ระบบดำเนินไปได้อย่างไม่มีปัญหา

4.6.2 จุดมุ่งหมายของ MPI

- สร้างส่วนอินเตอร์เฟส ที่สมบูรณ์ สามารถกระจายงานที่ เขียนมาเพื่อนทำงานบนเครื่อง ๆ เดียวไปยังเครื่องในระบบคลัสเตอร์ ถึงแม้ว่าในปัจจุบันการทำงานของ MPI จะทำงานได้ แต่ในขณะทำงานยังต้องทำงานผ่าน ไลบรารีของ MPI แล้วยังต้องมีการออกแบบโปรแกรมควบคุมการทำงานให้เป็นไปอย่างขนาน จึงจะทำให้การทำงานของระบบคลัสเตอร์เป็นไปได้อย่างมีประสิทธิผล
- สร้างระบบที่มีใช้การสื่อสารผ่านระบบเครือข่ายที่มีประสิทธิภาพ หลีกเลี่ยงการก๊อบปี้ระหว่าง หน่วยความจำด้วยกันเอง อนุญาตให้มีการเรียกใช้ข้อมูลและอ้าวอิงหน่วยความจำข้ามเครื่องได้ และพยายามให้การทำงานทั้งหมดตกไปอยู่ที่ ซีพียูที่ทำงานทางด้านคณิตศาสตร์ (Math Co-Processor) ทั้งหมด ซึ่งที่กล่าวมานี้สามารถใช้งานได้แล้วใจปัจจุบัน
- สร้างระบบที่สามารถใช้งานได้ร่วมในสภาวะแวดล้อมของระบบคลัสเตอร์ขนาดใหญ่
- สามารถควบคุมการส่งข้อมูลในระบบเครือข่ายอย่างมีประสิทธิภาพ โดยปราศจากปัญหาที่เกิดจากการสูญหายของข้อมูลและการผิดพลาดของข้อมูล

- สร้างระบบที่มีความเป็นกันเองไม่มีความซับซ้อนในการใช้งาน ทำให้การทำงานเสมือนอยู่ในเครื่อง ๆ เดียว โดยการควบคุมในเครื่องโหนดทั้งหมดสามารถควบคุมได้จากเครื่องกลางเครื่องเดียว ดังเช่นที่ PVM, NX, P4 ทำไว้
- สามารถสร้างอินเตอร์เฟสของระบบ ที่มีความเป็นมาตรฐานสามารถใช้งานร่วมบนเครื่องหลายๆสถาปัตยกรรมได้ โดยไม่ต้องแก้ไขใด ๆ เมื่อต้องการนำไปใช้งานบนเครื่องที่ใช้สถาปัตยกรรมอื่น

4.6.3 ภาษาที่สามารถนำมาใช้ร่วมกับ MPI ได้

MPI ได้พัฒนาให้ใช้งานร่วมกับ 2 ภาษาหลักที่ใช้โดยทั่วไปในโลกขอยูนิกส์ ได้แก่ ANSI C และ Fortran 77 โดยสามารถทำงานร่วมกับการเขียนโปรแกรมเชิงวัตถุได้

โดยมีแนวคิดที่จะนำ Fortran 90 และ C ++ มาพัฒนาให้สามารถนำมาใช้ร่วมกับ Fortran 77 และ ANSI C ได้ ถึงแม้ว่าคำสั่งของ Fortran 90 จะมีคำสั่งหลายชุดที่ไม่สามารถใช้งานบน Fortran 77 ได้ และรวมถึงโครงสร้างของ C++ มีหลายส่วนที่ไม่เหมือนกับ ANSI แต่แนวคิดนี้ก็เป็นแนวคิดที่เป็นไปได้ที่จะนำมาใช้ โดยทั้งหมดยังอยู่ระหว่างการพัฒนาให้นาสามารถนำมาใช้งานได้

4.7 โปรแกรมมอนิเตอร์

การมอนิเตอร์ระบบคลัสเตอร์นั้นมีข้อมูลหลาย ๆ อย่างที่จำเป็นต่อการใช้งาน การมอนิเตอร์ทั่วไปจะเป็นการมอนิเตอร์ สถานะว่าเครื่อง, รายละเอียดของเครื่องแต่เครื่อง และ ความสามารถในการควบคุมเครื่องในบางโอกาส

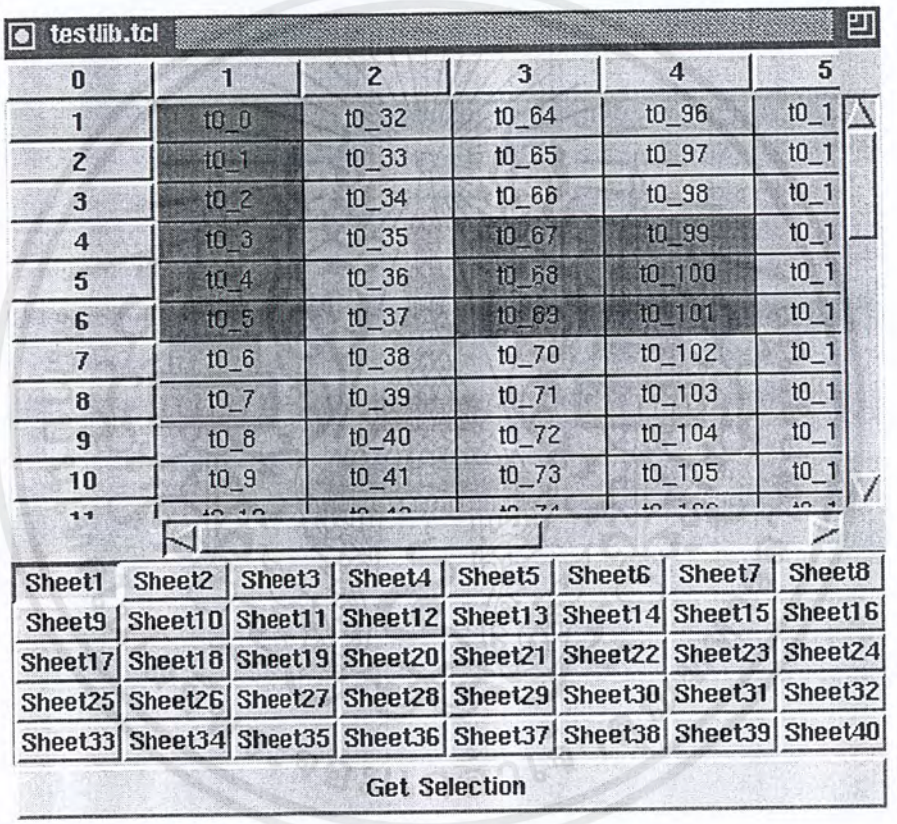
โดยตัวอย่างโปรแกรมมอนิเตอร์ที่ค่อนข้างสมบูรณ์ตามจุดประสงค์ของโปรแกรมมอนิเตอร์ คือ SCMS (Smile Cluster Management System) ที่ได้รับการพัฒนาขึ้นโดยภาควิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ โดย SCMS ได้ทำการพัฒนามาเพื่อใช้งานตามจุดประสงค์ของการมอนิเตอร์ โดย SCMS ออกแบบมาเพื่อให้ ผู้ดูแลระบบสามารถมอนิเตอร์ ข้อมูลต่าง ๆ สามารถ ส่งคำสั่งไปยังเครื่องแต่ละเครื่องได้ตามต้องการ สามารถเรียกขอข้อมูลจากเครื่องแต่ละโหนดได้ รวมถึงสามารถแก้ไขค่าติดตั้งบนเครื่องได้ และยังมีความสามารถอื่นๆอีกหลายด้าน SCMS เป็นโปรแกรมที่เขียนขึ้นให้มีการตอบสนองผู้ใช้ ด้วยกราฟิกทำให้สามารถมอนิเตอร์ และควบคุมคลัสเตอร์ที่มีจำนวนมาก ๆ ได้อย่างง่ายดาย ความสามารถของ SCMS ในการมอนิเตอร์ระบบ

- สามารถเรียกขอข้อมูลในแต่ละเครื่อง เช่น OS และข้อมูลต่างๆในเครื่อง
- สามารถดูโหลดรวมของแต่ละเครื่องเพื่อดูการทำงานของเครื่อง ความโหลดของเครื่อง
- มี API ในส่วนที่สามารถเข้าไปเรียกข้อมูลของเครื่องโหนดได้โดย สามารถใช้ API ผ่าน C , Java, และ Tcl/TK
- มีการติดต่อยังผู้ใช้และผู้ดูแลระบบ เป็นกราฟิกส์

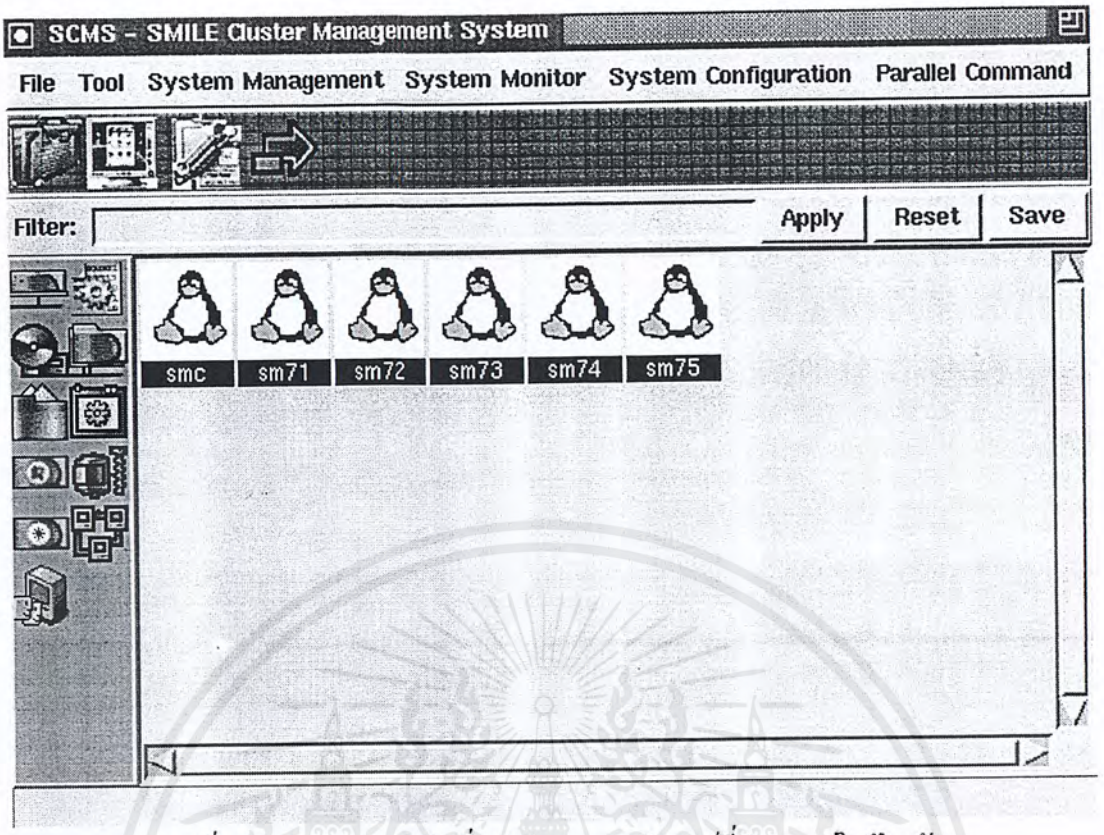
- การตรวจสอบข้อมูลต่าง โดยสามารถดู ดิสก์ที่เหลือ, ผู้ใช้งาน หรือสั่งให้เครื่องบูต หรือ ปิดได้ รวมถึงคำสั่งอื่นมากมาย
- มีการมอนิเตอร์ข้อมูลต่างๆเป็นแบบทันทีทันใด
- สามารถตรวจสอบและแก้ไขค่าติดตั้งต่างบนแต่ละเครื่องโหลดได้
- มีระบบเตือนโดยสามารถตั้งค่าตามความต้องการของผู้ใช้ว่าจะต้องการให้มีการเตือนเมื่อไร

- สถาปัตยกรรมของโปรแกรมได้ถูกเขียนให้มีการทำงานแบบ ไคลเอนต์-เซิร์ฟเวอร์

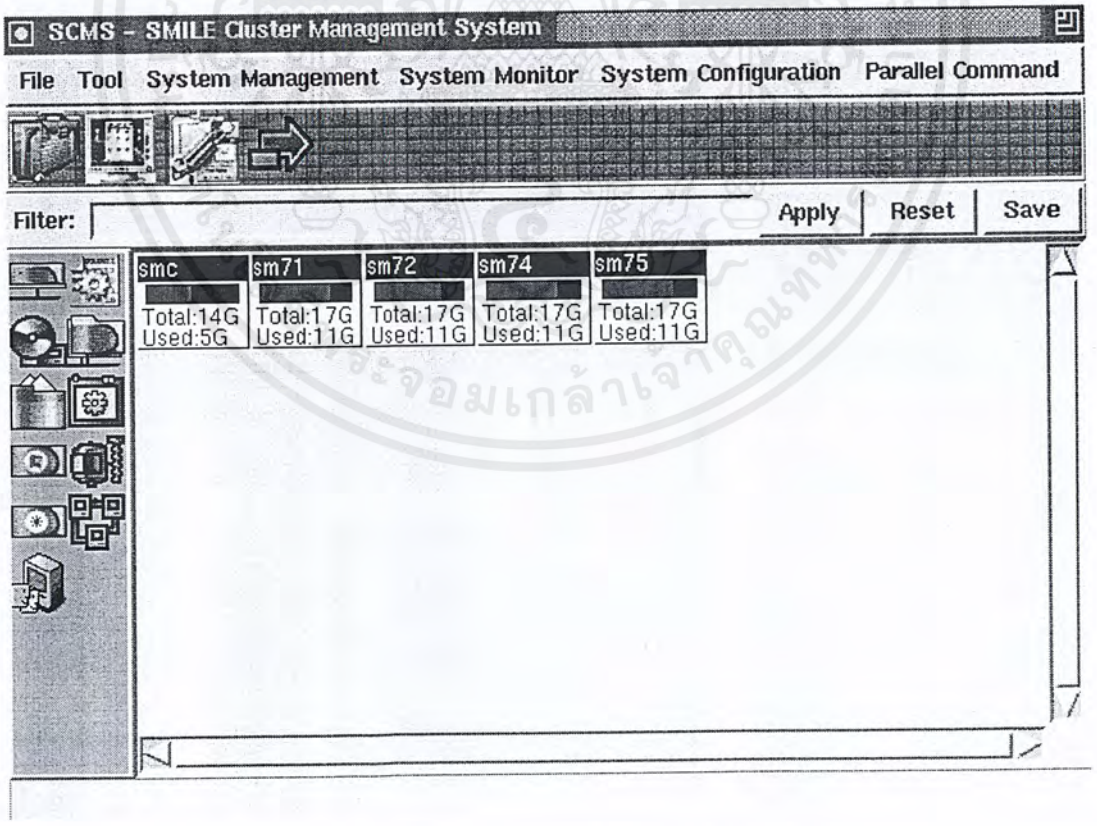
ตัวอย่างการมอนิเตอร์ระบบคลัสเตอร์โดยผ่านระบบ SCMS



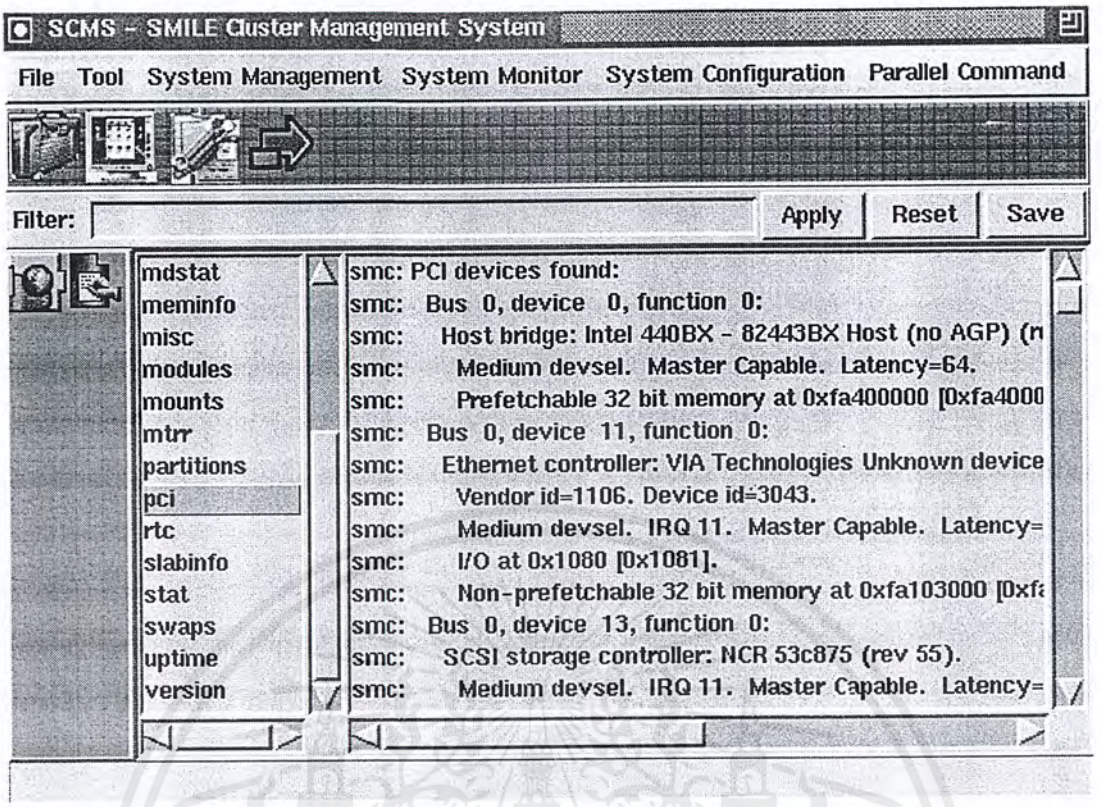
รูปที่ 4.5 แสดงเครื่องทั้งหมดที่อยู่ในระบบคลัสเตอร์



รูปที่ 4.6 แสดงสถานะว่ามีเครื่องใดในระบบคลัสเตอร์ที่ทำงานอยู่ในปัจจุบัน



รูปที่ 4.7 แสดงการใช้งานดิสก์ในแต่ละเครื่องโหนด



รูปที่ 4.8 แสดงการค่าติดตั้งในแต่ละเครื่องโหนด

4.8 โปรแกรมโมสิคซ์ (Mosix)

โปรแกรมโมสิคซ์เป็นซอฟท์แวร์ ซึ่งได้รับการออกแบบมาสำหรับเพิ่มประสิทธิภาพของเคอร์เนล ลีนุกส์เพื่อให้รองรับการทำงานแบบคลัสเตอร์ (Cluster computing) โดยเฉพาะ โดยโปรแกรมโมสิคซ์จะช่วยในการทำการกระจายโหลดให้สมดุลกัน (Load – balancing) และใช้อัลกอลิทึมเพื่อให้สามารถใช้หน่วยความจำ, อินพุท และเอาท์พุทได้อย่างมีประสิทธิภาพที่สุด ซึ่งจะขึ้นอยู่กับทรัพยากรของระบบคลัสเตอร์ที่มีอยู่ ยกตัวอย่างเช่น ถ้าเกิดการกระจายโหลดในแต่ละโหนดไม่เท่ากัน หรือถ้าโหนดใดโหนดหนึ่งเกิด disk swapping มากเกินไป ทำให้ไปหน่วงหน่วยความจำที่ว่างอยู่ของอีกโหนดหนึ่ง ในกรณีแบบนี้โปรแกรมโมสิคซ์จะเริ่มเคลื่อนย้ายโพรเซสจากโหนดหนึ่งไปอีกโหนดหนึ่ง เพื่อทำให้เกิดการสมดุลกันของโหลดในแต่ละโหนด หรือเคลื่อนย้ายโพรเซสไปยังโหนดที่มีหน่วยความจำว่างอยู่ หรืออาจจะลดการทำงานของ I/O

การทำงานของโปรแกรมโมสิคซ์จะมีลักษณะเป็นแบบ transparent กับโปรแกรมประยุกต์อื่น ๆ นั่นหมายความว่าเราสามารถจะเอ็กซิคิวต์โปรแกรมประยุกต์ที่เป็นแบบตามลำดับ (sequential) และแบบขนาน(parallel) ได้เหมือนกับเรากำลังทำในระบบแบบ SMP โดยเราจะไม่จำเป็นต้องสนใจว่า โพรเซสรันอยู่ที่ไหน เพราะว่าหลังจากที่เราสร้างโพรเซสการทำงานขึ้นมาใหม่ โปรแกรมโมสิคซ์จะหาโหนดที่ดี

ที่สุดในการทำงาน ณ เวลานั้น ๆ แล้วก็ให้โหนดที่เหมาะสมนั้นรับงานไปทำ โดยการทำงานแบบนี้ไม่จำเป็นต้องเปลี่ยนการเชื่อมต่อของลีนุกซ์ (Linux Interface) นั่นคือเราสามารถเห็นและควบคุมโพรเซสของเราทั้งหมดได้ ถ้าโพรเซสเหล่านั้นทำงานอยู่ในโหนดของเรา

อัลกอลิทึมของโปรแกรมโมสิคซ์จะเป็นรูปแบบของการกระจาย คือ แต่ละโหนดจะเป็นทั้งเครื่องที่สร้างโพรเซสใหม่ขึ้นมา หรือรับโพรเซสมาจากโหนดอื่นเพื่อทำงาน ซึ่งแต่ละโหนดนั้นสามารถเพิ่มเข้าหรือย้ายออกไปจากระบบคลัสเตอร์เมื่อไหร่ก็ได้ โดยจะรบกวนการทำงานของโพรเซสน้อยมาก และประโยชน์อีกอย่างหนึ่งของโปรแกรมโมสิคซ์ก็คือ อัลกอลิทึมในการมอนิเตอร์ ซึ่งทำให้เราทราบถึงความเร็วของแต่ละโหนด, โหลด, หน่วยความจำที่ว่างอยู่ และอัตราการใช้ I/O ของแต่ละโพรเซส โดยข้อมูลเหล่านี้ได้นำมาใช้ในการตัดสินใจเพื่อหาโหนดที่เหมาะสมในการทำงานของโพรเซสนั้น ๆ

ประโยชน์ที่ได้รับจากการใช้โปรแกรมโมสิคซ์

คุณลักษณะหลักของโปรแกรมโมสิคซ์ คือ การกระจายโหลดในสมดุลกัน และอัลกอลิทึมในการเคลื่อนย้ายโพรเซส ทำให้ผู้ใช้ไม่จำเป็นต้องมีความรู้เกี่ยวกับสเตจการทำงานของโหนด ณ เวลานั้น ๆ โดยมีประโยชน์อย่างมากในการทำ time – sharing, multiuser environment โปรแกรมประยุกต์แบบขนานสามารถเอ็กซิคิวต์ได้โดยการ fork หลาย ๆ โพรเซส เหมือนกับทำในระบบ SMP โดยโปรแกรมโมสิคซ์จะจัดสรรทรัพยากรให้สามารถทำงานได้อย่างมีประสิทธิภาพที่สุด

โปรแกรมโมสิคซ์จะมีประโยชน์สำหรับโปรแกรมประยุกต์และสภาวะ ดังต่อไปนี้

- **CPU – bound processes** การคำนวณที่ใช้เวลาในการทำงานนาน (มากกว่า 2 – 3 วินาที) เช่น การคำนวณทางวิทยาศาสตร์และวิศวกรรม หรือโพรเซสที่มีการผสมระหว่างเวลาในเอ็กซิคิวต์ (ทั้งเร็วและช้า) เราแนะนำให้ใช้ PVM / MPI ในการเริ่มต้นโพรเซส
- **Scalable Web servers** เป็นการทำงานที่มีการคำนวณเป็นจำนวนมากสำหรับ transaction ที่เข้ามา
- **มัลติยูสเซอร์, time – sharing environment** ผู้ใช้หลาย ๆ คนสามารถใช้ทรัพยากรร่วมกันได้ โปรแกรมโมสิคซ์มีประโยชน์กับผู้ใช้โดยจะทำการจัดสรรโพรเซสเซอร์ในการทำงานของโพรเซสใหม่ในลักษณะ transparent
- **Parallel processes, especially processes with unpredictable arrival and execution times** ระบบจัดการโหลดให้สมดุลแบบไดนามิกของโปรแกรมโมสิคซ์ สามารถแสดงผลการจัดการรายงานแบบคงที่ ผ่านทางการเอ็กซิคิวต์
- **I/O – bound and mixed I/O and CPU processes** โดยการเคลื่อนย้ายโพรเซสไปที่ไฟล์เซิร์ฟเวอร์
- **ระบบคลัสเตอร์ที่มีความเร็วและขนาดของหน่วยความจำไม่เท่ากัน** ระบบการจัดการทรัพยากรที่สามารถเปลี่ยนแปลงได้ของโปรแกรมโมสิคซ์จะทำให้ได้ประสิทธิภาพการทำงานที่ดีที่สุด

บทที่ 5
การใช้งานและทดสอบระบบ

จากจุดประสงค์ของหัวข้อโครงงานที่ศึกษา เพื่อให้ได้ความรู้ในการนำมาออกแบบระบบที่ตรงตามความต้องการที่จะสร้างระบบที่สามารถใช้งานได้ และอยู่บนจุดมุ่งหมายของโครงงานที่ต้องการสร้างระบบซูเปอร์คอมพิวเตอร์ราคาถูก แต่ต้องมีประสิทธิภาพในระดับที่เหมาะสม ดังที่กล่าวมาข้างต้น คือ เราจะนำระบบดิสก์เลสเข้ามาร่วมอยู่ในโครงสร้างหลักของระบบ โดยส่วนของระบบหลัก ๆ จะประกอบด้วย

- ดิสก์เลส
- NIS / NFS / RSH
- PVM / MPI / MOSIX
- Library ที่สนับสนุนการประมวลผลแบบขนาน

การทำงานของดิสก์เลสคลัสเตอร์ ต้องมีลำดับการบูทระบบที่ถูกต้องโดยการเริ่มเปิดระบบ จะต้องเริ่มเปิดที่เครื่องคลัสเตอร์คอนโทรลเลอร์ให้เรียบร้อย โดยที่เครื่องนี้จะต้องมีการเปิดบริการ bootp, nfs และอื่นๆ ตามที่ได้กล่าวมาแล้วข้างต้น เพื่อให้เครื่องไคลเอ็นต์สามารถขอเข้าใช้ไฟล์ที่จำเป็นต่อการบูทและการใช้งานได้ เมื่อเครื่องคลัสเตอร์คอนโทรลเลอร์เปิดบริการครบตามที่ได้กล่าวมาแล้ว จึงสามารถบูทเครื่องไคลเอ็นต์ได้ โดยระบบทดสอบจะบูทจากแผ่นดิสก์ที่ประพฤติตนเสมือนเป็นการบูทรอม หลังจากนั้นจึงเป็นการเสร็จสิ้นส่วนของการบูทระบบ

การทำงานของระบบคลัสเตอร์นั้น หลังจากที่ผู้ใช้ได้รับอนุญาตจากผู้ดูแลระบบให้ผู้ใช้ผู้นั้นสามารถเข้าใช้งานในระบบคลัสเตอร์ได้ ผู้ใช้จะได้ล็อกอินและรหัสผ่านที่จะใช้เพื่อเข้าสู่ระบบ เมื่อสามารถเข้าสู่ระบบได้แล้วก็ทำการเรียกใช้คำสั่งต่างๆที่ต้องการทำ โดยให้โปรแกรมที่รันและผลลัพธ์ที่ต้องการให้ออกมาอยู่ที่โฮมไดเรกทอรีของตน โดยระหว่างการสั่งงานนั้น เมื่อผู้ใช้ได้สั่งให้ระบบทำงานใดๆ เช่น สั่งให้ pvm คำนวณเวกเตอร์ เครื่องคลัสเตอร์คอนโทรลเลอร์จะประมวลผลงานที่ต้องแบ่งและจ่ายงานไปยังเครื่องโหนด โดยผ่านเดมอนของตนเอง ถ้าเป็น pvm ก็คือ pvmd หรือถ้าเป็น lam/mpi ก็คือ lamd โดยการเข้าใช้งานเครื่องโหนดนั้น เครื่องคลัสเตอร์คอนโทรลเลอร์จะใช้ rsh เข้าไปสั่งงาน โดยผ่านล็อกอินของผู้ใช้ผู้นั้นเข้าไป ดังนั้นโพรเซสที่รันอยู่ที่เครื่องโหนดทุกเครื่อง จะเป็นโพรเซสของผู้ใช้ผู้นั้น

<u>โครงสร้างของระบบทดสอบ</u>

ระบบที่ได้ทำการทดสอบเป็นเครื่องเพนเทียมทู 400 Mhz จำนวน 5 เครื่อง โดย มีหน่วยความจำรวมทั้งหมด 320(64x5) เมกะไบต์ โดยอยู่ที่เครื่องคลัสเตอร์คอนโทรลเลอร์ จำนวน 64 เมกะไบต์ ซึ่งเท่า

กับเครื่องไคลเอ็นต์ทุกเครื่องที่มี 64 เมกะไบต์เท่าๆกัน โดยแต่ละเครื่องติดตั้งเน็ตเวิร์กการ์ด ขนาด 10 Mbps โดยเครื่องทั้งหมด ลงระบบปฏิบัติการลินุกส์ slackware

การทดสอบระบบที่ได้สร้างขึ้นมาทดสอบ จะจำลองการเข้ารหัสไฟล์เพลง (.wav) ให้เข้าอยู่ในรูปแบบไฟล์แบบ MP3 โดยใช้โปรแกรม bladeenc ซึ่งพัฒนาให้สามารถทำงานบนระบบที่ทำงานแบบขนานได้ โดยโปรแกรม bladeenc นี้สามารถติดตั้ง และใช้งานร่วมกับ lam/mpi

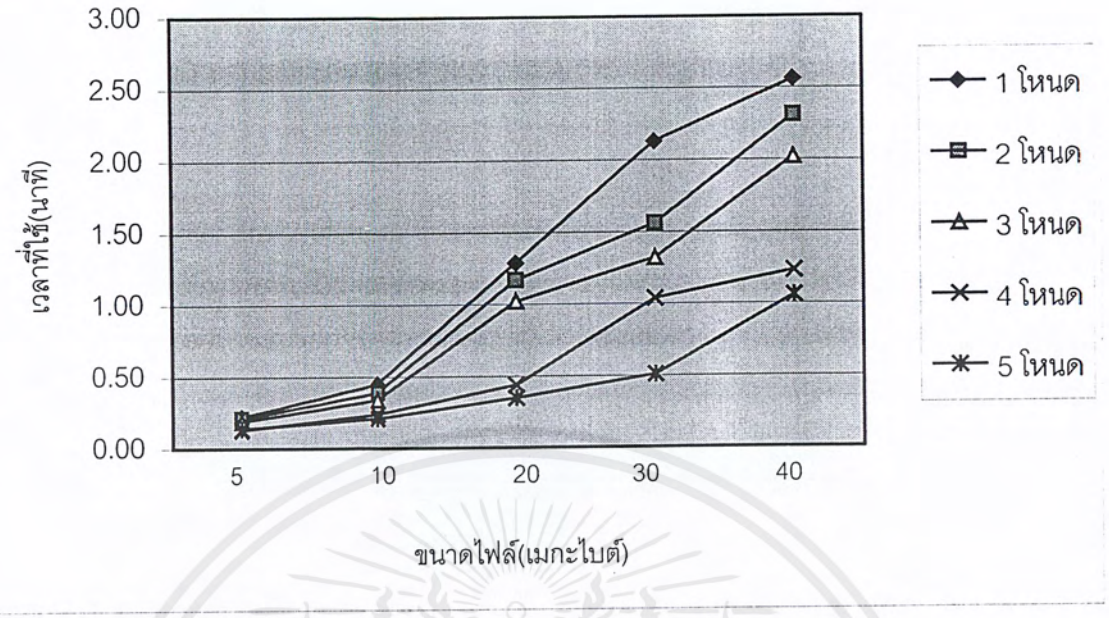
การใช้งาน สามารถใช้งานโดยผ่าน MPI โดยการเรียกใช้งาน ทำได้ดังนี้

- การใช้งานบน MPI ทำโดยใช้คำสั่งในการกระจายงานไปยังเครื่องโหนดต่าง ๆ ตัวอย่าง เช่น ต้องการกระจายงานไปยังเครื่องต่าง ๆ จำนวน 4 เครื่อง โดยใช้โปรแกรม bladeenc เข้ารหัสเพลงจากไฟล์ 123.wav เป็น 123.mp3 จะเรียกคำสั่งดังนี้ที่คอนโซลของยูนิกส์ (/usr/local/lam/bin/mpirun –np 4 /usr/local/bin/bladeenc 123.wav 123.mp3)

จากการใช้งานระบบคลัสเตอร์ของทั้งสองโปรแกรม ได้มีการทดสอบประสิทธิภาพของการทำงานของระบบคลัสเตอร์ โดยมีการเพิ่มลดจำนวนเครื่องโหนดในระบบเป็นจุดเปลี่ยนผันแรก และใช้ขนาดของไฟล์ที่เข้ารหัสเป็นจุดเปลี่ยนผันที่สอง โดยผลการทดสอบระบบได้ผลดังนี้

ขนาดไฟล์(เมกะไบต์) / จำนวนโหนด(เครื่อง)	5	10	20	30	40
1	0.22	0.45	1.29	2.13	2.56
2	0.21	0.39	1.17	1.56	2.31
3	0.19	0.34	1.02	1.32	2.02
4	0.13	0.24	0.44	1.04	1.23
4+1	0.14	0.21	0.35	0.51	1.06

ตารางที่ 5.1 แสดงผลการทดสอบระบบด้วยโปรแกรม bladeenc โดยผ่าน MPI



รูปที่ 5.1 กราฟแสดงการเปรียบเทียบ ประสิทธิภาพของการทดสอบระบบ แบบ MPI

บทที่ 6
วิจารณ์และสรุปผลจากการทำงานของระบบ

การทำงานของระบบ SuperComputing Clustering ที่กล่าวมาทั้งหมดนั้น เป็นระบบที่ทดสอบเพื่อการนำมาใช้ทดแทนระบบซูเปอร์คอมพิวเตอร์ที่ใช้อยู่ในปัจจุบัน ถึงแม้ว่าการทำงานของระบบดังกล่าวจะยังมีปัญหาเกี่ยวกับความเสถียรของระบบอยู่บ้าง ดังเช่น ถ้าหากมีโหนดใดไม่สามารถทำงานต่อได้ อาจจะทำให้การทำงานในขณะนั้นล่มไปทั้งระบบได้

จากการทดสอบเห็นได้ว่าถึงแม้ระบบจะทำงานได้จริง แต่ว่าการทำระบบคลัสเตอร์นั้นขึ้นกับหลายอย่างประกอบรวมกัน ทั้งทางด้าน ซอฟท์แวร์และฮาร์ดแวร์ โดยจะต้องมีส่วนประกอบทั้งสองที่พอเหมาะกัน จึงจะทำให้ได้ประสิทธิภาพที่ดี แต่จากเครื่องที่เราทดสอบในครั้งแรกนั้นด้อยในหลาย ๆ ด้าน ทำให้การทดสอบจึงไม่เห็นผลมากนัก ผู้ทดสอบจึงได้พยายามปรับเปลี่ยนค่าติดตั้งต่าง ๆ บนเครื่องทดสอบ แต่ก็ไม่ทำให้ประสิทธิภาพเพิ่มตามที่ควรจะเป็น

จากที่ได้กล่าวในข้างต้นว่า คำว่าโครงงานนี้มุ่งเน้นในคำว่าประหยัด แต่ผลลัพธ์ที่ได้มากลับไม่ได้มาซึ่งประสิทธิภาพที่ดีตามต้องการ จึงแสดงให้เห็นว่า คำว่าประหยัด อาจส่งผลเป็นอันมากต่อผลการทำงานของระบบ การที่เราประหยัดโดยไม่คำนึงถึงสิ่งใดๆเลยอาจทำให้เกิดความสูญเสียได้ จากเครื่องที่เราได้มา เพื่อนำมาสร้างระบบครั้งแรกนั้น ส่วนประกอบเป็นเพียงเครื่อง 486DX66 6 เครื่อง โดยมีหน่วยความจำรวมเพียง 72 เมกกะไบต์เท่านั้น ถ้าหากมองย้อนกลับไปถึง ระบบคลัสเตอร์ของ Beowulf ที่เป็นจุดเริ่มต้นของระบบคลัสเตอร์ จะเห็นได้ว่า ระบบนั้นประกอบขึ้นด้วย 486DX4 ทำงานที่ความถี่ 100Mhz และมีหน่วยความจำในแต่ละโหนด ถึง 16 เมกะไบต์ ซึ่งมากกว่า ระบบที่เราทดสอบถึง 2 เท่า แล้วเครื่องคลัสเตอร์คอนโทรลเลอร์ ยังมีประสิทธิภาพสูงกว่าที่กล่าวมาอีก จากโครงสร้างของเครื่องที่เป็นดังกล่าวทำให้พอจะสรุปให้เป็นสาเหตุที่ทำให้ประสิทธิภาพของเครื่องไม่ดีขึ้นเท่าที่ควรจะเป็น

แต่เมื่อผลการทำงานของระบบคลัสเตอร์ที่เราสร้างขึ้นจากเครื่องที่เหลือใช้นั้นนำมาซึ่งประสิทธิภาพที่ไม่ดีกว่าเดิมมากนัก ถึงแม้จะได้พยายามปรับแต่งค่าต่าง ๆ ที่อาจทำให้เกิดข้อผิดพลาดในการทดสอบให้เหลือน้อยที่สุดเท่าที่จะเป็นไปได้แล้วก็ตาม แต่ก็ยังเป็นไปตามที่ได้กล่าวมาแล้วข้างต้น ทางกลุ่มโครงงานจึงได้พยายามหาระบบที่นำมาทดสอบแทน เพื่อให้เห็นถึงประสิทธิภาพของการทำงาน โดยนำเครื่องที่มีประสิทธิภาพสูงขึ้นมาอีกระดับมาใช้ทดสอบประสิทธิภาพและการทำงานของระบบแทน โดยใช้เครื่องเพนเทียมทู 400MHz จำนวน 5 เครื่องมาร่วมในการทดสอบ โดยการทดสอบระบบนี้ ได้ทำให้เครื่องทั้งหมดอยู่ในระบบปิด เหมือนระบบแรก โดยผลการทดสอบที่ได้จากระบบใหม่ ดังรูป 5.1 นั้น จะเห็นได้ว่า ประสิทธิภาพของระบบในการทำงานเข้ารหัส MP3 มีแนวโน้มที่จะสูงขึ้น เป็นสัดส่วนที่คงที่เมื่อเพิ่มจำนวนเครื่อง

จากภาพรวมการทดสอบที่ได้กล่าวมาจนถึงการทดสอบ และผลการทดสอบ ทำให้ทราบถึงสาเหตุหลายประการที่มีผลต่อการทำงานของระบบดิสก์เลสคลัสเตอร์ ถ้าหากสามารถเลือกอุปกรณ์ และควบคุมสภาวะต่าง ๆ ให้เหมาะสม ก็จะทำให้ระบบดิสก์เลสคลัสเตอร์ทำงานได้อย่างมีประสิทธิภาพ และเป็นซูเปอร์คอมพิวเตอร์ราคาประหยัดที่สามารถนำมาใช้งานได้จริง

บรรณานุกรม

[1] Rajaraman V. (1993) : "*Supercomputers*", Wiley Eastern Limited ,1978.

[2] Kai Hwang (1993) : "*Advanced Computer Architecture : Parallelism, Scalability, Programmability*", McGraw-Hill, Inc. International Editions

[3] Michael Hord R. (1940) : "*Parallel Supercomputing in MIMD Architectures*", CRC Press, Inc.

[4] William Stallings (1996) : "*Computer Organization and Architecture : Designing for Performance*", Prentice Hall Fourth Edition

[5] http://www.extremelinux.org/

[6] http://www.slug.org.au/NIC/index.html

[7] http://www.beowulf.org/howto/bootp.html

[8] http://smile.cpe.ku.ac.th/

[9] http://bond.imm.dtu.dk/jobd/

[10] http://epm.ornl.gov/pvm/

[11] http://lam.mpi.nd.edu/lam

[12] http://www.mcs.anl.gov/mpi/mpich

[13] http://www.mpi.nd.edu/lam/software/xmpi

[14] http://www.netlib.org/lapack

[15] http://www-unix.mcs.anl.gov/dbpp/tools.html

[16] http://www.pdl.cs.cmu.edu/NASD/Cheops_pfs.html

[17] http://www.mpi.nd.edu/~jsquyres/bladeenc/

[18] http://www.mosix.org